(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 July 2001 (26.07.2001)

PCT

(10) International Publication Number
WO 01/54344 A1

(51) International Patent Classification[7]: H04L 9/00, G06F 13/00, H04L 9/00, 9/08, 9/32

(21) International Application Number: PCT/US01/02143

(22) International Filing Date: 22 January 2001 (22.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/488,345	20 January 2000 (20.01.2000)	US
09/488,337	20 January 2000 (20.01.2000)	US
09/488,143	20 January 2000 (20.01.2000)	US
09/488,613	20 January 2000 (20.01.2000)	US
09/488,155	20 January 2000 (20.01.2000)	US
09/489,600	20 January 2000 (20.01.2000)	US
09/488,614	20 January 2000 (20.01.2000)	US
09/489,601	20 January 2000 (20.01.2000)	US
09/489,596	20 January 2000 (20.01.2000)	US
09/499,247	7 February 2000 (07.02.2000)	US

(71) Applicant: INTERACTUAL TECHNOLOGIES, INC. [US/US]; 100 Century Center Court, #205, San Jose, CA 95112 (US).

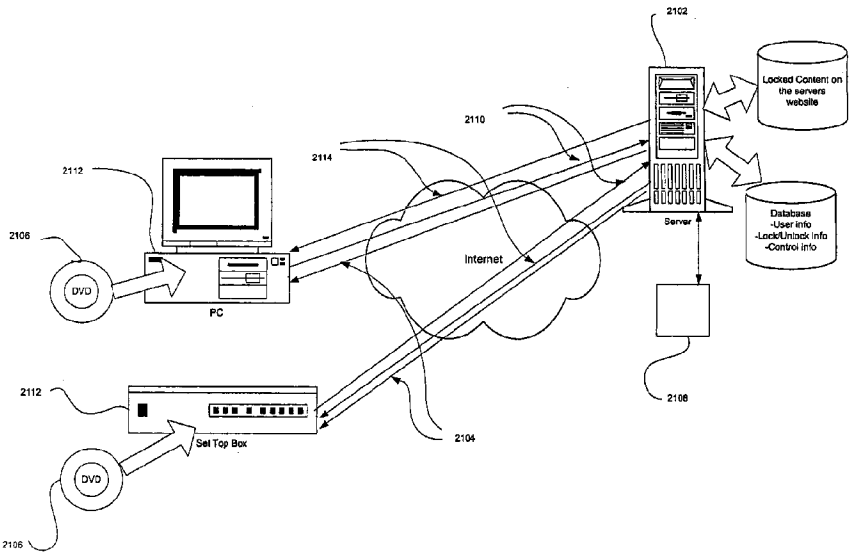
(72) Inventors: COLLART, Todd, R.; 206 Arbuelo Way, Los Altos, CA 94022 (US). LAMKIN, Allan, B.; 4282 Farley Lane, San Diego, CA 92122 (US). GETSIN, Evgeniy, M.; 11556 E. Powers Avenue, Englewood, CO 80111 (US). LEWIS, Michael, J.; 2058 South Kenton Court, Aurora, CO 80014 (US).

(74) Agents: RANNEY, Kathleen, A. et al.; Fitch, Even, Tabin & Flannery, Suite 1600, 120 South LaSalle Street, Chicago, IL 60603 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR,

[Continued on next page]

(54) Title: SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR REMOTE CONTROL AND NAVIGATION OF LOCAL CONTENT



(57) Abstract: A system, method and article of manufacture are provided for remotely controlling content stored on a local computer connected to a network system such as the Internet. The embodiment allows content such as DVD (2106) Video content to be locked so that play can only be accomplished through verification of user identity and also allows augmentation and supplementation of the content provided by the video from a remote server (2102). In addition, upon verification of user client credentials, the present embodiment allows the locally stored content to be supplemented with additional content delivered over the network system. Furthermore, the present embodiment allows content stored locally on a client device (2112) to be remotely controlled to play synchronously on a plurality of client devices.

WO 01/54344 A1

LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR REMOTE CONTROL AND NAVIGATION OF LOCAL CONTENT

BACKGROUND

Multimedia computer systems have become increasingly popular over the last several years due to their versatility and their interactive presentation style. A multimedia computer system can be defined as a computer system having a combination of video and audio outputs for presentation of audio-visual displays. A modern multimedia computer system typically includes one or more storage devices such as an optical drive, a CD-ROM, a hard drive, a videodisc, or an audiodisc, and audio and video data are typically stored on one or more of these mass storage devices. Audio and video data for a multimedia display may also be stored in separate computer systems that are networked together. In this instance, the computer system presenting the multimedia display would receive a portion of the necessary data from the other computer system via the network cabling.

Such audio and video data or content is often stored on media such as CD-ROM or digital video disc (DVD). However, once a vendor has delivered such content to a customer, the vendor loses any practical control over the product. Even if the product is delivered under license rather than out right sale, it has traditionally been difficult to prevent a customer from copying the content or providing the content to any number of friends so that they might illegally copy the content.

Another problem which arises from the vendors loss of control of the content maintenance and updating of the software. If content is to be added or modified, the vendor must send a new disc to the customer. In addition, the vendor can not control the amount of data which the customer can access. In other words, once the disc is delivered, the customer will have access to all of the content on the disc and only that content. Time sensitive content, such as advertising, can become obsolete but will still be accessible on the disc.

Therefore, there remains a need for a system method or apparatus allowing flexible control of content delivered to a client. Such a system, method or apparatus would preferably allow content to be initially delivered on a traditional recording medium such as a CD-ROM or DVD but would allow a vendor to remote control the

access of a user to the content stored thereon. Furthermore, such a system would preferably allow a vendor supplement and/or modify the content and could allow the vendor to limit a client's access to certain portions of the locally stored content if desired. Furthermore, remote control of navigation would be preferably and could facilitate simultaneous access by a controlled number of multiple clients if desired.

SUMMARY OF THE INVENTION

A system, method and article of manufacture are provided for synchronizing an event on a plurality of a client apparatuses. First, an event is stored in a memory storage device. The client apparatuses and a host computer are adapted to be connected to a network. In operation, information is transmitted from the host computer to the memory storage device utilizing the network. This allows for the simultaneous playback of the event on each of the client apparatuses.

In a another further embodiment, the invention can be characterized as a system, method and article of manufacture for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses. Initially, an event is stored in memory on at least one of a plurality of client apparatuses. These client apparatuses and a host computer are adapted to be connected to a network during use. Information is stored on the host computer for allowing the simultaneous playback of the event on each of the client apparatuses. In operation, the information may be downloaded utilizing the network for playback after the simultaneous playback of the event from the memory.

In a supplemental embodiment, the present invention can be characterized as, a system, method and article of manufacture for providing overlays during a synchronized event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network. Further, an event may be simultaneously played back on the client apparatuses utilizing the network. During the playback of the event, visual and/or audio material may also be overlaid on the event based on input received utilizing the network.

In an additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for delayed synchronization of an event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network and an event is stored in memory on the client apparatuses.

The event is then simultaneously played back on the client apparatuses utilizing the network. During the simultaneous playback, a request may be received from one of the client apparatuses for that particular client to be included in the synchronized event. In response to the request, information is transmitted to the requesting client apparatus utilizing the network. This information is adapted for identifying a location in the memory where the event is currently being played back on the remaining client apparatuses. This allows the simultaneous playback of the event on the requesting client apparatus. In one embodiment of the present invention, the request may be received utilizing the network.

In another embodiment, the present invention can be characterized as, a system, method and article of manufacture for synchronizing an event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network. Next, an application program is embedded on a site on the network. In use, information is requested from a server on the network utilizing the application program. Such information relates to an event to be played back simultaneously on the client apparatuses. In response to such request, a script is received for displaying the information.

In a further additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses. First, a request is received utilizing a network for viewing an event. Next, the request is queued in memory. In response to the request, an object is created which is adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal. The object is sent to one of the client apparatuses utilizing the network for being stored therein.

In yet another embodiment, the present invention can be characterized as, system, method and article of manufacture for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses. First, various values are determined including a current time, a start time when an event is to start, and a stop time when the event is to end. Thereafter, a length of the event is calculated based on the start time and the stop time. If any portion of the length of the event takes place during a predetermined

threshold period, a command is stored in memory. Further, a loop is created at the start time during which a lapsed time of the event is tracked.

In still yet an additional embodiment, the present invention can be characterized as a system, method and article of manufacture for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses. First, a plurality of events are stored in memory on a plurality of client apparatuses. The events each have a unique identifier associated therewith and which are stored in the memory. In operation, the client apparatuses are adapted to be coupled to a host computer via a network. The identifier of the event which is stored in the memory of the client apparatuses is then retrieved utilizing the network. Such identifier is subsequently compared with an identifier of a scheduled event. If the comparison renders a match, the playback of the event is begun on each of the client apparatuses.

In a further additional embodiment, the present invention can be characterized as, a system, method and article of manufacture are for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event. A type of the playback device of each of the client apparatuses is first identified. A command associated with the identified type of the playback device is then looked up in a table. Thereafter, the command is sent to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

In another additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for remotely controlling local content for local access and use by a client device. An input is received from the client which can allow a transaction server to identify the client. Once the client has been identified a command can be sent to the client which controls the manner in which the client device can use and access the local content.

The local content can be embodied on a digital video disk, and commands generated by the transaction server can be in the form of an unlock sequence which allows the client device to access and use the content stored on the disk. In addition, commands from the transaction server can be used to navigate the content stored on the disk and can even supplement the content stored thereon. The transaction server

can, in response to a client identification, unlock content stored remotely on the transaction server, allowing content to be easily maintained and updated remotely at a single transaction site without having to replace many DVD disks being used by many different clients.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood when consideration is given to the following detailed description thereof. Such description makes reference to the annexed drawings wherein:

Figure **1** is a schematic diagram of a hardware implementation of one embodiment of the present invention;

Figure **2** illustrates a flowchart delineating a method for synchronizing an event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **3** illustrates a flowchart delineating a method for storing synchronization information for subsequent playback of an event in accordance with one embodiment of the present invention;

Figure **4** illustrates a flowchart setting forth a method for providing overlays during a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **5** illustrates a flow diagram for delayed synchronization of an event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **6** illustrates a flow diagram for providing information on a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **7** illustrates a method for creating a synchronizer object in order to playback an event simultaneously on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **8** illustrates a flowchart for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses in accordance with one embodiment of the present invention;

Figure **9** is a flowchart delineating a method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses in accordance with one embodiment of the present invention;

Figure **10** shows a flowchart delineating a technique for interfacing a plurality of different types of playback devices of client apparatuses which are networked to simultaneously playback an event in accordance with one embodiment of the present invention;

Figure **11** illustrates the manner in which a layer factory is created in accordance with one embodiment of the present invention;

Figure **12** illustrates the manner in which user requests are processed in accordance with one embodiment of the present invention;

Figures **13-16** illustrate various class/component diagrams in accordance with one embodiment of the present invention;

Figure **17** illustrates a logical sequence diagram in accordance with one embodiment of the present invention;

Figure **18** illustrates a logical sequence diagram that shows server side collaboration in accordance with one embodiment of the present invention;

Figure **19** illustrates a logical sequence diagram showing client side collaboration in accordance with one embodiment of the present invention;

Figure **20** is a schematic diagram of a process according to an embodiment of the present invention;

Figure **21** is a schematic diagram illustrating an embodiment of the present invention;

Figure **22** is a schematic diagram illustrating another embodiment of the present invention;

Figure **23** is a schematic diagram illustrating yet another embodiment of the present invention;

Figure **24** is a schematic diagram illustrating still another embodiment of the present inveniton;

Figure **25** is a flowchart illustrating a method for carrying out the present invention; and

Figure **26** is a flowchart illustrating a method for carrying out an aspect of the present inveniton.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figures **1-26** illustrate a system for remotely controlling content stored locally on a client apparatus. Prior to use, an event is stored in memory on at least one client apparatus. Such client apparatus is adapted to be connected to a network along with a host computer(s). In operation, information is transmitted from the host computer to the at least one client apparatus utilizing the network. This information controls playback of the event stored on locally on the client computer.

In various embodiments, the client apparatus may take the form of a computer, television, stereo, home appliance, or any other types of devices. In one embodiment, the client apparatuses and the host computer each include a computer such as an IBM compatible computer, Apple Macintosh computer or UNIX based workstation.

A representative hardware environment is depicted in Figure **1**, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **110**, such as a microprocessor, and a number of other units interconnected via a system bus **112**. The workstation shown in Figure **1** includes a Random Access Memory (RAM) **114**, Read Only Memory (ROM) **116**, an I/O adapter **118** for connecting peripheral devices such as disk storage units **120** (i.e. DVD playback device) to the bus **112**, a user interface adapter **122** for connecting a keyboard **124**, a mouse **126**, a speaker **128**, a microphone **132**, and/or other user interface devices such as a touch screen (not shown) to the bus **112**, communication adapter **134** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **136** for connecting the bus **112** to a display device **138**. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present embodiment may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using Java, C, and the C++ language and utilizes object oriented programming methodology. A preferred embodiment of the

embodiment utilizes HyperText Markup Language (HTML), including Java, JavaScript (ECMAScript), ActiveX, or the like, to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the server. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H, Frystyk, T. Berners-Lee, J. Gettys and J.C. Mogul, "Hypertext Transfer Protocol -- HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML is an application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

Synchronization Overview

Figure **2** illustrates a flowchart delineating a method for synchronizing an event on a plurality of client apparatuses. First, in operation **200**, an event is stored in memory on at least one of the client apparatuses. In various embodiments, the memory may take the form of an electromagnetic medium, or any type of optical storage device, i.e. CD-audio. In a primary aspect of the present embodiment, the memory includes a digital video disc (DVD) (audio or video). Further, for reasons that will soon become apparent, the information includes chapter information associated with the DVD. In such embodiment where the memory is portable, the user may be required to purchase the memory, i.e. DVD, in order to participate in a synchronized event, thus increasing the sale of DVD's.

It should be noted that the event need not be necessarily stored in memory on all of the client apparatuses, but rather stored on one or some of the client apparatuses and streamed to the remaining client apparatuses at variant rates. This may be feasibly accomplished if the client apparatus(es) containing the stored event has a high-bandwidth connection with the remaining client apparatuses. For example, the client apparatus(es) containing the stored event may include a server that has a connection to a plurality of televisions via a cable network, i.e. WEBTV. Similar functionality may be achieved via a broadcast medium. The present

embodiment is thus flexible by having an ability to host user events and cooperative events.

In one embodiment, the event includes a video and audio presentation such as movie, a concert, and/or a theatrical event. It should be noted, however, that the event may included any recording capable of being played back for entertainment, education, informative or other similar purposes.

In use, the client apparatuses and a host computer are adapted to be connected to a network. Such network may include a wide, local or any other type of communication network. For example, a wide area network such as the Internet may be employed which operates using TCP/IP or IPX protocols.

In operation **202**, information is transmitted from the host computer to the appropriate client apparatuses utilizing the network. This information allows for the simultaneous and synchronous playback of the event on each of the client apparatuses. In one embodiment, the information may also include a start time when the playback of the event is to begin on each of the client apparatuses. Further, an ending time may be included when the playback of the event is to end on each of the client apparatuses. Still yet, "play" command information may be sent to the client apparatuses at the start time. As an option, input may be received from the user, and used to alter the playback of the event. The host server, or synchronization server, can also control various streams of a variant rate and different hardware associated with those streams.

The present embodiment thus has the ability to synchronize video playback for one or multiple (thousands) users from one or multiple physical locations, and to synchronize with external video, audio and/or data streams.

Users of the present embodiment are at multiple physical locations and host servers may also be at different locations. The present embodiment is thus a scalable system which is capable of servicing an unlimited number of users. Since the content is local to the user machine, no high network bandwidth is required.

History Download Capabilities

Figure **3** illustrates a flowchart delineating a method for storing synchronization information for subsequent playback of an event. Initially, in

operation **300**, an event is stored in memory on at least one of the client apparatuses, as set forth earlier. These client apparatuses are adapted to be connected to a network along with a host computer during use.

In operation **302**, information is stored on the host computer(s) for allowing the simultaneous playback of the event on each of the client apparatuses. In one embodiment, the information may include a history and data associated with the synchronous playback. In particular, the history may include any overlaid material(as will be described hereinafter in greater detail), any specific commands affecting the playback of the information, or any other type of general information, i.e. start time, end time, etc.

In operation **304**, the information may be downloaded utilizing the network at any time after the synchronous playback of the event. Such downloaded information may then be used for playback after the simultaneous playback of the event. As such, the present embodiment has the ability to allow users to download a history and data associated with a particular synchronization event and play it later.

Overlay Synchronization

Figure **4** illustrates a flowchart setting forth a method for providing overlays during a synchronized event on a plurality of client apparatuses or any other source. First, in operation **400**, a plurality of client apparatuses are connected via a network. In operation **402**, an event may be simultaneously played back on the client apparatuses utilizing the network, as set forth earlier.

During the playback of the event, visual and/or audio material may also be overlaid on the event based on input received from at least one of the client apparatuses. See operation **404**. This may be accomplished by transmitting the overlay material from one of the client apparatuses to the host computer or any other server, and multicasting the same to the remaining client apparatuses.

As an option, the overlay material may include annotations on a display of the client apparatus. For example, the overlay material may include sketches which are inputted by way of a stylus-based input screen or a keyboard or the like, along with a voiceover inputted by way of a microphone or voice synthesizer. Such capability may also be quite valuable in an educational environment.

In one embodiment, the overlay material may also be displayed on each of the client apparatuses utilizing the network. This allows each of the users to experience the overlay in real-time during the simultaneous playback of the event. As an option, the user inputting the overlay material may select which users may experience the overlay material. The client apparatus that provided the overlay material may also be identified to the users experiencing the overlay material.

It should be noted that various bi-directional communication may be enabled for allowing data to travel to and from the server. For instance, the playback of the event on the client apparatuses may be altered in any feasible way based on input from a user.

Late Synchronization

Figure **5** illustrates a flow diagram for delayed synchronization of an event on a plurality of client apparatuses. First, in operation **500**, a plurality of client apparatuses are connected via a network and an event is stored in memory on the client apparatuses. The event is then simultaneously played back on the client apparatuses utilizing the network, as set forth earlier. Note operation **502**.

During the simultaneous playback, a request may be received from one of the client apparatuses for that particular to be included in the synchronized event, as set forth in operation **504**. This request may be received after the synchronized event has already begun while it is still playing. Further, the request may be submitted via a site on a network, i.e. website.

In response to the request, information is transmitted in operation **506** to the requesting client apparatus utilizing the network. This information is adapted for identifying a location in the memory where the event is currently being played back. This allows the simultaneous playback of the event on the requesting client apparatus.

The end users are thus able to come in at a later time and to be synchronized with the event. Targeted synchronization and various filters criteria can be applied to target different audiences. Also language and cultural differences can be taken into account. Still yet, the present embodiment may be adapted to address users on different hardware platforms (MAC, PC, set-top boxes). This may be accomplished

by identifying the user using a cookie, a user profile which is identified by way of a log in, or a Burn Cut Area (BCA) of the disc.

An example setting forth details relating to identifying DVDs will now be set forth. First, a content owner (such as studio) requests use of the BCA on their DVDs. Based on request, the replicator (examples include WAMO, Panasonic, Nimbus, Technicolor, Pioneer, Crest) adds unique BCA number to every DVD. Adding BCA number to each DVD requires a special (YAG) laser. This may be the very last step in the manufacturing process. The BCA numbers for a specific DVD must then be entered into InterActual's BCA database. Information to track includes: DVD title, i.e. "Lost in Space"; BCA #/range, i.e. 12345687890; and Shipping Packaging/Tracking Container, i.e. Box 52221 to Hollywood Video.

After the BCA number is added to the DVDs, the DVDs are packaging/boxed for distribution to either the Distributor or the Retailer. It should be noted that many companies take multiple forms, so the replicator and distributor may be one in the same. Also, some retailers are large/important enough to get shipments directly from replicator. The way in which the DVDs are packaging/shipped is very important because one must track the BCA numbers to actual shipping containers (box, etc.). Therefore tracking information must also be added to the BCA database.

If packaged DVDs are then sent to distributor, the distributor also has mechanisms, i.e. scanners, input device, and monitoring devices, in place for tracking based on their distribution. For example, Deluxe may receive a "package" of 100,000 copies of "Lost in Space." However, the distributor ships 10,000 to Retailer A and 5,000 to Retailer B. The distributor should be able to "input" retailer A and B's distribution information into the system. Ideally, this becomes a seamless/automated process.

Once the DVDs reach the retailer (either from the replicator or distributor), then DVDs may be further divided and distributed to local stores/outlets. In such a situation, the retailer should be able to automatically "track" distribution of these DVDs through to their stores. Over time, all three entitities (replicator, distributor, and retailer) are able to add tracking information to BCA database. Due to complexity and dependencies on existing business systems, the retail tracking concept will be rolled out in phases: replicator first most likely with key retail

accounts. The distributors will be brought in. Retailers will then begin to embrace the ability to track based on local outlet/store.

By the foregoing design, easy deployment is thus afforded and minimal hardware is required to allow the synchronization of content without significant capital investments and with a very efficient control mechanism. The content delivery does not rely on high network bandwidth and is independent from the synchronization.

Internet Server Application Program Interface (ISAPI) extensions will be used on the server. ISAPI extensions provide a mechanism to maintain a temporary or permanent connection with the users. These connections allow the Synchronization Server to process request and to send the appropriate DVD commands. The permanent connections are known as "Keep Alive" connections. ISAPI extension can also be used as an HTTP interface to a more traditional server, with all data returned as text.

On the client side the approach is to use, but not limited to Java 1.1 applets, to initiate event start-up for the Synchronization server. The advantage of using Java 1.1 applets is to achieve platform independence for existing and future Java-enabled devices. JavaScript will be used to provide user interface navigation by "wrapping" the applet.

An ISAPI (Internet Server Application Program Interface) is a set of Windows program calls that let one write a Web server application that will run faster than a Common Gateway Interface (CGI) application. A disadvantage of a CGI application (or "executable file," as it is sometimes called) is that each time it is run, it runs as a separate process with its own address space, resulting in extra instructions that have to be performed, especially if many instances of it are running on behalf of users. Using ISAPI, you create a Dynamic Link Library (DLL) application file that can run as part of the Hypertext Transport Protocol (HTTP) application's process and address space. The DLL files are loaded into the computer when HTTP is started and remain there as long as they are needed; they don't have to be located and read into storage as frequently as a CGI application.

Existing CGI applications can be converted into ISAPI application DLLs without having to rewrite their logic. However, they do need to be written to be thread-safe so that a single instance of the DLL can serve multiple users.

A special kind of ISAPI DLL is called an ISAPI filter, which can be designated to receive control for every HTTP request. One can create an ISAPI filter for encryption or decryption, for logging, for request screening, or for other purposes.

One can write ISAPI server extension DLLs (ISAs) that can be loaded and called by the HTTP server. Users can fill out forms and click a submit button to send data to a Web server and invoke an ISA, which can process the information to provide custom content or store it in a database. Web server extensions can use information in a database to build Web pages dynamically, and then send them to the client computers to be displayed. An application can add other custom functionality and provide data to the client using HTTP and HTML.

One can write an ISAPI filter. The filter is also a DLL that runs on an ISAPI-enabled HTTP server. The filter registers for notification of events such as logging on or URL mapping. When the selected events occur, the filter is called, and one can monitor and change the data (on its way from the server to the client or vice versa). ISAPI filters can be used to provide custom encryption or compression schemes, or additional authentication methods.

Both server extensions and filters run in the process space of the Web server, providing an efficient way to extend the server's capabilities.

Overall Component Design

The various functional components of the software associated with the present embodiment will now be set forth. Such components include a Java/JavaScript Component, Synchronizer Component, LayerImpl Component, Business Layer Component, Configuration Manager Component, and DBConnect Component.

Java/JavaScript Component

Figure **6** illustrates a flow diagram for providing information on a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present embodiment. First, in operation **600**, a plurality of client

apparatuses are connected via a network, as set forth earlier. Next, an application program is embedded on a site on the network in operation **602**. Such application program may take the form of a JAVA applet, and the site may include a website on the Internet.

In use, information is requested from a server on the network utilizing the application program. See operation **604**. Such information relates to an event to be played back simultaneously on the client apparatuses and may include general information such as a start and stop time of the event, or more specific information about the event itself.

In response to such request, a script is received for displaying the information. Note operation **606**. The script may take any form such as Perl, REXX (on IBM mainframes), and Tcl/Tk, and preferably includes a JAVAscript.

In one variation of the present embodiment, the JAVA applet may be further adapted to send a request to retrieve command information from the server for use with a playback device of one of the client apparatuses. The commands may be adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses. Further, the commands may include a start time when the playback of the event is to begin on each of the client apparatuses.

The JAVA applets and JAVAscript are used to communicate with the playback device of the client apparatuses. In one embodiment, the playback device includes a PCFriendly TM video player manufactured by Interactual ®.

The Java applet is embedded within a web page and uses HTTP protocol to communicate to the synchronization server. The applet could request event information from the server, and display it to the user via JavaScript. The applet could also send a "*BroadcastVideoEvent*" request to retrieve DVD commands that can be passed to the video component, as set forth hereinabove.

Synchronizer Component

Figure 7 illustrates a method for creating a synchronizer object in order to playback an event simultaneously on a plurality of client apparatuses. The synchronizer object is portion of the software that actually implements the

synchronization procedure. First, in operation **700**, a request is received utilizing a network for viewing an event. Next, the request is queued in memory in operation **702**.

In response to the request, in operation **704**, an object is created which is adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal. As an option, the activation signal may be provided using a clock of the client apparatus, or located at a different location, i.e. server. To accomplish this, the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

In operation **706**, the object is sent to one of the client apparatuses utilizing the network for being stored therein. In accordance with a primary aspect of the present embodiment, the object may be adapted to playback the event which is stored in memory of the client apparatus. This may be accomplished by activating a digital video disc (DVD) player.

In summary, when the Synchronizer component receives a "*BroadcastVideoEvent*" from the applet, it then places the request in the thread queue for processing. To process a request, the thread creates a "call back" object, if one does not exist for this event. The thread then adds the request to the "call back" object queue. This "call back" object will be invoked when it is time to play the DVD. The Synchronizer component creates a Call Back COM object, *LayerSink*. The Synchronizer component is also responsible for creating the *LayerFactory* interface which will be set forth hereinafter in greater detail.

LayerImpl Component

Figure **8** illustrates a flowchart for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses. The present method ensures that critical information is tracked during the synchronization of the event. Such critical information not only ensures proper synchronization, but also enables various peripheral features.

First, in operation **800**, various values are determined including a current time, a start time when an event is to start, and a stop time when the event is to end.

Thereafter, a length of the event is calculated based on the start time and the stop time in operation **802**. As an option, the current time is determined by querying a clock of one of the client apparatuses.

If any portion of the length of the event takes place during a predetermined threshold period, a command is stored in memory in operation **804**. The command may be adapted to automatically begin playing back the event at the start time. In one embodiment, the threshold period includes the time the users can be queued before the event. As an option, chapter information may be stored in the memory if any portion of the length of the event takes place during the predetermined threshold period. This allows the command to automatically begin playing back the event at a predetermined chapter.

In operation **806**, a loop is created at the start time during which a lapsed time of the event is tracked. This information may be used for various tracking purposes to decide when to issue commands to the user. In another embodiment, a second loop may be created upon the beginning of a chapter during which information on a next chapter is retrieved.

The "call back" object (*LayerSink*) is thus responsible for creating and communicating with the *LayerImpl* component. The *LayerImpl* component acts as a scheduler, determining when to issue commands to the user.

LayerImpl will issue different DVD commands, based on the type of decoder the user has in their PC. *LayerImpl* will differentiate between the decoders by using the decoder information submitted from the client. The *LayerImpl* will pass the correct DVD command to the client, based on the decoder's capabilities. For example, if the decoder does not support the *TimePlay* event, then the server may send a *ChapterPlay* event and wait appropriately.

The following is an enumerated summary of the steps the component uses to determine when the users will receive the DVD commands:

1. Retrieves the current time, and the time the event starts and ends.
2. Calculates the length of the event.
3. If the event is within a threshold period (**i.e.** the time users can be queued before the event), then store the first DVD command in memory. Also, store the Chapter information in memory.
4. Create a loop that processes request until the event has completed.

5. In the loop, calculate the lapsed time of the event.
6. In the loop, retrieve the next chapter information.
7. Create another loop that will loop until time for the next chapter to be played.
8. When the next chapter is ready to play, send the command that was retrieved from the Chapter table.

Business Layer Component

Figure **9** is a flowchart delineating a method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses. This features is important since a host server may be synchronizing more than one event at once, or during overlapping times. Such events must therefore be distinguished.

First, in operation **900**, a plurality of events are stored in memory on a plurality of client apparatuses. Each of the events is assigned a unique identifier which is stored in the memory.

In operation **902**, the client apparatuses are adapted to be coupled to a host computer via a network, as set forth hereinabove. In operation **904**, the identifier of the event which is stored in the memory of the client apparatuses is then retrieved utilizing the network. Such identifier is subsequently compared with an identifier of a scheduled event, as set forth in operation **906**. If the comparison renders a match, the playback of the event is begun on the appropriate client apparatuses. Note operation **908**.

CbusinessLayer thus differentiates events by the disk and location ids, uploaded by the client to guarantee backwards compatibility. As set forth earlier, late arrivals can always re-sync with the event.

Configuration Manager Component

Figure **10** shows a flowchart delineating a technique for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event. The present technique is important since the playback devices of

the various client apparatuses may differ in make and model. Thus, different commands are required therefor.

In operation **1000**, a type of the playback devices of the client apparatuses is first identified. Such "type" may refer to a make, model, or any other distinguishing characteristic of the particular playback devices. A command associated with the identified type of the playback device is then looked up in a look-up table. Note operation **1002**. Such table may be located at the host server, or at any other location such as the client apparatuses.

Thereafter, in operation **1004**, the command is sent to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

This component is thus responsible for identifying what type of reference player is hosting the event. The reference player can be the database, which contains the DVD commands or a real time player. When the initial DVD is command is requested, the "Synchronizer" table is queried for the host type. From that point forward, the scheduler would know from whom to receive data.

DBConnect Component

This component is responsible for communicating with the Synchronizer tables, and for providing access methods for the retrieved data. All interaction from the tables is on a read-only basis. The *LayerImpl* component communicates with this component to retrieve DVD commands and event information.

Even though current implementation may be based on a Microsoft platform, hard dependencies on Microsoft or any other 3rd-party development tools may be avoided. To address such issues, the following considerations may be made throughout the code:

MFC specific code may be avoided. Instead, STL may be used. ATL and/or MFC code may be encapsulated into separate classes and portioned from the rest of the code. Class implementations may use aggregation pattern to delegate business logic to the portable classes. Database connection classes may be separated and the communication protocol may be separated with respect to portability to Oracle and other platforms.

Figures **11** and **12** illustrate the order of events among the various components of the present embodiment. In particular, Figure **11** illustrates the manner in which a layer factory is created. As shown, an event is first checked in a database server after which a business layer is created in a WEB server in a manner set forth hereinabove. The foregoing components are then created. Figure **12** illustrates the manner in which user requests are processed. As shown, communication is afforded with the video player on the client machine by means of JAVAscript and JAVA applets. The WEB server, in turn, communicates DVD commands to the video player via the JAVA applets, and also interfaces the database server via the various components thereof which were set forth hereinabove.

Alternate Embodiments

To support future enhancements, further components may be included with extendibility as the major objective. Various future enhancements of the product and how they will be addressed will now be set forth.

Hosted Real Time Players

While spirals may retrieve pre-recorded DVD commands from the database, alternate spirals may support a consumer as a host. The architecture may also support plug-in components. Alternate spirals may support the RealTimeConnector component, which accepts host user request and forwards them to the clients. The instant architecture supports the DBConnector which accepts events from the database.

Keep Alive Connections

Clients may maintain connections throughout the event. This allows the host to send a various number of commands to the client of the event. Although the spiral disconnects users once a PLAY command has been issued, the Synchronizer class (which will be set forth later) adds each connection to a Thread Pool. This pool of connections can be left open during the life of the event.

Logging Participants

Each request may be logged into the database to provide a reference for the future.

DVD Positioning

As an option, connections may be pooled to allow the synchronization server to direct consumer's machines to the certain locations throughout the entire event.

Synchronization events in alternate spirals may be defined as a combination of play from location event and the actual event. This way, one describes each event in the unambiguous way on the client side and synchronizes it with the server. For example, a situation may be considered where one fast forwards after a movie is played for 15 min and thereafter plays the scene in the movie. In such situation, one has to submit the information to the client player, indicating that it (player) has to start time play from 15 min into the movie and fast-forward to the certain location. A better way would be to analyze what is the next event after fast forwarding occurred and perform a combination for the play from location and next event. This design would require significant changes to the client infrastructure, including *video object*, *remoteagent* and *provider* and should be taken into consideration in any alternate client design.

Classes/Component Diagrams

Figures **13-16** illustrate various class/component diagrams. In particular, Figures **13-16** illustrate a Synchronizer Class Diagram **1300**, LayerImpl Class Diagram **1400**, Business Layer Class Diagram **1500**, and DBConnect Class Diagram **1600**, respectively.

Sequence Diagrams

Figure **17** illustrates a logical sequence diagram **1700**. As shown, when the server receives a user request, it analyzes the authentication information of the request (date/time, disc id, user id, and BCA number) and the appropriate synchronization event stored in the database. The database contains an *event start threshold* value measured in milliseconds. This threshold defines the amount of time prior to an event that a consumer is eligible to "connect" for the start of the event.

If the date/time of the user request lies within the event start threshold, the user is put into wait queue and receive the appropriate data when the time elapses. Note steps 1,2,3,5,6,7 of the Logical Sequence diagram. Otherwise, a message is sent informing the user when the event will occur. Note step 4 of the Logical Sequence diagram.

Server side collaboration diagram

Figure **18** illustrates a logical sequence diagram **1800** that shows server side collaboration. As shown, server ISAPI extension receives a BroadcastVideoEvents request. It calls *IA_BusinessServer* via *BeginProcess,* to retrieve configuration information. Configuration information contains a playback connector. Playback connector identifies whether the server will have to communicate with a reference player or will it perform playback from the database.

At step 6, ISAPI extension will call *IA_BusinessServer CompareTime* method and based on the results will send to the user a predefined web page indicating to retry later or return control to the web server, notifying it (web server) to keep the connection open. At this point connection is pooled and will be processed by the *IA_BusinessServer* at a time of the event.

Client Collaboration Diagram

Figure **19** illustrates a logical sequence diagram **1900** showing client side collaboration in accordance with one embodiment of the present embodiment.

Classes/Interfaces Definition

Definitions of one embodiment of the various classes associated with the software which implements the present embodiment will now be set forth.

Class Applet1

Purpose:

This is the class that implements the applet. The browser will use it to bootstrap our applet.

Responsibilities:

- Request a BroadCastVideo event and to gather event status information.

Collaborations:

BroadCastEvent, CITIEncrypt

Base class and implemented interfaces:

Javax.Applet

Public interface:

getChapter Returns the current chapter the reference player is playing.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getTitleInfo Returns the current title the reference player is playing
Return type: String

Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTime Returns the time the event is scheduled to start <SS:MM:HH:DD:MM:YYYY>
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeSec Returns the time the event starts in seconds.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeMinReturns the time the event starts in minutes.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeHrReturns the time the event starts in Hours.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

GetStartTimeDay Returns the time the event starts in days.
Return type: String
Parameters: void
Pre-conditions: None.

Post-conditions: None.

GetStartTimeMnth Returns the time the event starts in months.

Return type: String

Parameters: void

Pre-conditions: None.

Post-conditions: None.

GetStartTimeYr Returns the time the event starts in year.

Return type: String

Parameters: void

Pre-conditions: None.

Post-conditions: None.

GetLenOfEvent Returns the length of the event.

Return type: String

Parameters: void

Pre-conditions: None.

Post-conditions: None.

GetExpiredTime: Returns lapse time of the event.

Return type: String

Parameters: void

Pre-conditions: None.

Post-conditions: None.

getServerTime: Returns the servers current time <SS:MM:HH:DD:MM:YYYY>.

Return type: String

Parameters: void

Pre-conditions: None.

Post-conditions: None.

getServerTimeSec: Returns the servers current in seconds.

Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeMin: Returns the servers current in minutes.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeHr: Returns the servers current in hours.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeDay: Returns the servers current in day.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeMnth: Returns the servers current in month.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeYr: Returns the servers current in year.
Return type: String
Parameters: void
Pre-conditions: None.

Post-conditions: None.

startProc: Calls the ISAPIs "ServerInfo" method.
Return type: void
Parameters: String disk id, String location id
Pre-conditions: None.
Post-conditions: None.

msgEvent: Calls BroadCastEvent applet.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class BroadCastEvent

Purpose:

This is the class that invokes the Synchronizer.

Responsibilities:

- Sets the JavaScript with the command returned from the server.

Collaborations:

CITIEncrypt

Base class and implemented interfaces:

Java.Thread

Class CDBConnect

Purpose:

This is the class provides a public interface for components to request information from the DB tables.

Responsibilities:

- Opens the database and Synchronizer, Chapter_Disk tables.
- Queries the Synchronizer by the specified disk id and location id.
- Queries the Chapter_Disk by disk id.
- Provides the next chapter that is scheduled to play.
- Queries the Decoder_Capabilities table to determine if the requested player is time or chapter play.

Collaborations:

DBSyncSet
DBReferenceSet
CDBChapterSet
CDecoderCapabilities

Base class and implemented interfaces:

Public interface:

Get_NextChapter: Returns the next chapter to play.
Return type: String
Parameters: long time, long title, BSTR Chapter
Pre-conditions: None.
Post-conditions: None.

chkEvent: Checks if an event is scheduled for the disk and location id.

Return type: String
Parameters: long time, long title, BSTR Chapter
Pre-conditions: None.
Post-conditions: None.

get_initialDVDCommand: Returns the first DVD command to play.
Return type: String
Parameters: BSTR &
Pre-conditions: None.
Post-conditions: None.

get_nextDVDCommand: Returns the next DVD command to play.
Return type: String
Parameters: BSTR &
Pre-conditions: None.
Post-conditions: None.

decoderArray: Returns an array of decoder types.
Return type: String
Parameters: long **, long **
Pre-conditions: None.
Post-conditions: None.

Class CCConfigMgrImpl

Purpose:

This is the class provides a public interface for components to determine the type of reference player hosting the event.

Responsibilities:

- Opens the database and Synchronizer, Chapter_Disk tables.

- Queries the Synchronizer by the specified disk id and location id.
- Stores the reference player type.

Collaborations:

CConfigMgrRecSet

Base class and implemented interfaces:

Public interface:

get_hostType: Returns the reference player host type.
Return type: String
Parameters: short
Pre-conditions: None.
Post-conditions: None.

Class threadFunctor

Purpose:

This class provides a threading model that classes can use to derive.

Responsibilities:

- Calls the CreateEvent function, which opens a named or unnamed event objec.
- Calls _beginthread, which creates a thread begins execution of a routine at start_address. The routine at start_address must use the __cdecl calling convention and should have no return value. When the thread returns from that routine, it is terminated automatically.

- Calls the WaitForSingleObject function, which checks the current state of the specified object. If the object's state is nonsignaled, the calling thread enters an efficient wait state.
- Calls the ResetEvent function, which sets the state of the specified event object to nonsignaled.
- The state of an event object remains nonsignaled until it is explicitly set to signaled by the SetEvent or PulseEvent function.

Collaborations:

CConfigMgrRecSet

Base class and implemented interfaces:

Public interface:

start: Starts the thread.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

stop: Stops the thread. Calls CloseHandle for the thread and event.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class isapithread

Purpose:

This creates an ISAPI thread.

Responsibilities:

- Adds a request to a vector.
- Creates the sink object.
- Stores the request into sink object.
- Sends the time information to JavaScript.

Collaborations:

LayerSink
factorySink

Base class and implemented interfaces:

threadFunctor

Public interface:

addrequest: Adds the request to its vector.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getBLayerInfo: Responsible for getting information about the event.
Return type: void
Parameters: std:string&,std::string&, ChttpServerContext*
Pre-conditions: None.
Post-conditions: None.

Class factorySink

Purpose:

Manages the layerSink and businessLayerProp objects.

Responsibilities:

- Stores a layerSink object.
- Returns the "businesssLayerProp" <Business Layer Properties>
- Creates the "businessLayerProp" <Business Layer structure>

Collaborations:

LayerSink
businessLayerProp

Base class and implemented interfaces:

Public interface:

construct: Stores a layerSink object..
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

notifyCreateLayer: Responsible for creating a "businessLayerProp".
Return type: void
Parameters: BSTR, BSTR, DATE, DATE, LONG
Pre-conditions: None.
Post-conditions: None.

Class layerSink

Purpose:

layerSink represents a sink interface and stores a queue of requests. It creates a connection point object.
This call back object, allows asynchronously processing.

Responsibilities:

- Acts as the client sink object.
- Sends the results to the user
- Creates the "BusinessLayer" and makes it a connection point object.
- Closes the users connection.
- Creates a Factory interface by calling "createFactory".
- Creates a connection point for the factory.
- Stores the LayerSink in the FactorySink object.
- Creates a connection point (call back) by calling AtlAdvise, between the connection point container and the client sink object. This allows the client to receive events.
- Calls the connectable objects "getServerLayer". This method fires an event to the clients sink object.
- Create a business layer,
- Store the request in its vector.
- Release the Sink Object (client)
- Calls AtlUnadvise to terminates the ability of the client to receive events.

Collaborations:

Base class and implemented interfaces:

Public interface:

construct: Creates a connection point.
Return type: void

Parameters: void
Pre-conditions: None.
Post-conditions: None.

addRequest: Adds the request to its vector.
Return type: void
Parameters: BSTR, BSTR, DATE, DATE, LONG
Pre-conditions: None.
Post-conditions: None.

createBusinessLayer: Creates a business layer. Create the connection point .
Return type: void
Parameters: businessLayerProp &
Pre-conditions: None.
Post-conditions: None.

updatetime: This call back function translates the time and sends the command to the user.
Return type: void
Parameters: long,long
Pre-conditions: None.
Post-conditions: None.

Class CBusinessLayer

Purpose:

Creates a layerthread object. This object is responsible for providing access methods, which provide event information.

Responsibilities:

- The "Synchronizers" createBusinessLayer method creates a class object from the "IBusinessLayer" interface. <The class object is part of the LayerImpl project>
- The BusinesLayers class object <m_ilayer> calls its "Initialize" method. <Note: m_ilayer is the connection point object. It identifies the "Sink Interface".
- It then calls the "Initialize" method of the connection point.
- The "Initialize" method then calls the "ChkValidEvent" method, which then creates a layerthread object.

Collaborations:

CBusinessLayer
layerthread

Base class and implemented interfaces:

Public interface:

Initialize: Calls the "ChkValidEvent" method which kicks of a layer thread.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class layerthread

Purpose:

This object acts as a scheduler, processing request from its queue.

Responsibilities:

- Send DVD commands to the user.
- "Syncs" up late comers to the events.

Collaborations:

CBusinessLayer
CDBConnect

Base class and implemented interfaces:

Public interface:

startThread: Processes requests from the queue
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class CLayerFactory

Purpose:

This object manages businesslayer objects. Business layer objects communicate with the reference player and notify the user which DVD command to play.

Responsibilities:

- Send DVD commands to the user.
- "Syncs" up late comers to the events.
- This object Implements the IID_LayerFactory interface.
- This COM object is the servers Connectable Point object.

- This server object supports connections to sink interfaces. These sink interfaces reside on the client side and are equivalent to the "call back" functions in Windows.

Collaborations:

CBusinessLayer
CDBConnect

Base class and implemented interfaces:

Public interface:

getServerLayer: "Fires" an event to create a business layer with the properties retrieved from the pipe object.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

put_set_layer: call the "CLayerFactoryImpl" add() method. Supplying the "businesslayer" object.
This will added to shared memory queue and written to a file.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

FinalConstruct: Calls the "CLayerFactoryImpl" FinalConstruct COM class object.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

REMOTE CONTROL OF LOCAL CONTENT:

With reference to Fig. 20, the present embodiment provides a system and method for remote control of local content which enables the control of Video Playback from a remote server. Content stored on a medium 2002 such as a DVD is loaded onto a client device 2004. This hardware can be, for example, a computer, set top device such as is used to access WebTV, or some other device. The hardware device 2004 of the present embodiment has software 2006 in the form of a browser or presentation engine. In addition, the hardware 2004 has DVD Firmware or a Navigator 2008 in communication with the browser/presentation software 2006.

With continued reference to Fig. 20, a server 2010 delivers content to the hardware 2004 to be used in conjunction with the DVD 2002. This content can be in the form of ROM/HTML Content 2012 and/or DVD-Video Content 2014. Depending on the desired application, this content 2012 and 2014 enhances and/or allows a DVD experience 2016 provided by the DVD medium 2002.

With reference to FIG. 21, this control is performed by a transaction sever 2102 which sends video playback commands 2104 such as play stop, FF, Rewind, etc. It can also provide a locking/unlocking scheme which allows content on a local disk 2106 or website to be protected and accessible to particular users at prescribed points in time through a locking and unlocking process. This locking/unlocking technology could be broken down into two possible embodiments. For example, one such embodiment allows for unlocking local content that is on a local disk (i.e. DVD Disc) 2106 based on a user profile for example. In addition this content access could also expire or be accessible only during a particular time frame. Another possible embodiment allows for unlocking content on a website 2108 by requiring the user to have a DVD Disc in his computer's disc drive or set top box. (Therefore the user had to purchase the disk to get access to the on-line content).

This locking and unlocking is accomplished through the transaction server 2102, which validates the credentials of the user. These credentials 2110 are passed from the client 2112 (PC or set top box) and the server returns for example the unlock sequence 2114 to the client. In the case of DVD Video this unlock sequence can be in the form of General Purpose Registers Values (GPRM Bits) that unlock

the content. In the case of the website the client 2112 could pass the current disc 2106 in the drive's unique disc ID or BCA number and the transaction server 2102 allows a redirection to protected content after validation takes place. This unlocked content could be local or on the website 2108.

The advantages of remote control of a client's video device from a server is that the content can be protected. Since the information to use it is stored remotely, it can be easily maintained and upgraded, and allows introduction of new products without affecting the already shipped content (DVD Video). In addition, the control can be of a single client or many users. For example, with the ability to unlock content you can allow content to be accessible at a particular point in time thus allowing a special "event" or promotional time to occur and also allowing for various advertising/promotional models. The concept of expiring content is also useful, for example if an offer is only valid till the end of the year. The user is not burdened with viewing advertising or offers that they cannot participate in anyway after they have expired.

Another example is to reward users for purchasing particular products or even registering their products. The user can then be provided with additional content that is unlocked on the disc. In addition we can verify that people have the correct credentials before accessing content. To explain this further there may be website material that should only be accessible to customers who have purchased a particular DVD. The website may have additional information, games, or special items to be purchased and offered and this should only be made available to people who have purchased the DVD product. This may also span across multiple products for example if a user has purchased all of the available Lethal Weapon ™ titles it may be desirable to give additional content to that user for having purchased the series. Another embodiment of the embodiment follows the DIVX DVD model wherein a client is charged on a per usage basis for the content.

The present embodiment can also provide for remote navigation of content on a local server. For example navigation commands 2104 (Fig. 21) for left, right, up, down can be sent from the server to set General Purpose Registers (GPRMs) in the DVD Player that allowed content to be unlocked and viewed by users during the event. In addition, DVD navigation commands can be sent through streamed audio

with embedded triggers that send DVD navigation commands that call the video object in the web page.

In addition, With reference to Fig. 22, a synchronization server 2202 can be used to send commands to initiate video play in synchronization with several users 2112. Control can be of one or multiple clients in the form of PCs 2204 or set top players 2206. The remote navigation commands 2208 allow the server to tell the client what to do. The same set of commands can be sent to each of the clients – thus synchronizing the viewing experience. They could also be different, for example each user could be viewing a different DVD and therefore experiencing a different set of content. Based on the users profile 2210 they can also have access to different content. Given a geographical location or native language the control maybe tailored accordingly. The control of the video can be as simple as play, stop, fast forward, rewind, etc. or can include advanced features such as pan, zoom, rotate, etc. The type of navigation/control can be divided into 3 aspects:

- **Commands**. Commands control the playback and search mechanisms of a DVD-Video disc. (These can be issued from the server or the client)
- **Properties**. Properties are used to query attributes of the DVD-Video and set certain configuration properties. (These can be queried from the server or the client). An example is to get the current state or title/chapter/time of the DVD Video). Another example is to tell the user whether or not he or she is on a menu.
- **Events**. Events are used to trigger notification of various playback conditions, such as time changes, title changes and UOP changes. Events are essential for scripting and synchronizing the video with other assets. (these are sent from the content or client back to the server. They can indicate that the video has stopped playing, you are on a menu item, or the number to angles available in the video has changed.

Embodiment in a web page

With reference to FIG. 23, an embodiment of this embodiment provides control of content through a web page 2302. Using a browser as the client interface

(Such as Internet Explorer) the user can browse to the page 2302 on-line that contains an Active-X Control that has an embedded video object 2304. The client sends identification information and/or requests 2305 to the web page over the Internet 2306. In response, the video object 2304 activates video navigation commands to play and/or unlock sequences 2308 required to play the video.

With reference to FIG. 24, in another embodiment of the embodiment the Video Object 2304 opens a secondary connection 2402 to the transaction server 2102 to receive commands 2404 from the server to execute.

In another form the browser interprets http commands for the control of video. The web page on the server is viewed by the client and when the user selects an item in the web page the HTTP link can be formatted with parameters for the browser to interpret directly for video playback.

For example http://iti_video? command=Play

This is interpreted by the browser since it has an iti_video in the url and then parses the parameters, which in this case is the command to play.

Examples of embedding a Video Object in a Web Page

DVD-Video can be embedded within a HTML page and control its layout. Computer operating systems can embed DVD-Video using currently available embedding techniques. By way of example, each of the major computer operating systems is provided below:

Operating System	**Example**
Windows	<object ID=" Video Object" CLASSID="clsid:A0739DE5-571F-11D2-A031-0060977F760C" BORDER="1" WIDTH=50% HEIGHT=60% > </object>
Apple/Macintosh	<embed ID=" Video Object" TYPE="application/x- Video Object -plugin"

	ALT=" Video Object Plug In" HIDDEN="TRUE" > </embed>
Linux	TBD
Others	TBD

After the DVD-Video object is embedded in the web page, it can be accessed using any style sheet, link, or scripting language. Values for the IDstring must begin with a letter (A-Z or a-z) and may be followed by any number of letters, digits, hyphens, and periods up to a maximum of 48.

Unlike computers, set-top boxes do not generally have a full-blown operating system and browser. Therefore, the capabilities within the browser are often more restricted. For embedding DVD-Video within these platforms using ITX, the "Video Object" ID must be integrated within the embedded browser as any other tag structure. With this approach, any embedded browser that encounters the "Video Object" tag, would automatically associate this identifier.

Unlocking Implementation:

Another embodiment of the embodiment provides a system and method for unlocking portions of DVD-Video based on certain criteria (date, profile, BCA, etc.). To control playback of video, the video can be "locked" so that the consumer must perform certain steps to access and play the video. The steps that trigger the unlock should be controlled by the content owner and can be based on date, consumer profile, BCA number or any other criteria, and should be controllable over the Internet from a remote server. (Although it is possible to store the unlock sequence locally as well). With reference to FIG. 25, a method 2502 is provided for controlling content. The method 2502 begins with a step 2504 wherein a user tries to play a portion of DVD-Video from application or web page. In a step 2506, video software initiates a secure connection to a transaction server that authenticates the user, and then passes the correct unlock sequence of events back to the video software. This video software is preferably stored locally on the user's computer or, more preferably, on the disk on which the video content is stored. In a step 2508,

the events are passed from the server, and then in a step 2510 are then passed by the video software directly to the underlying hardware or software DVD decoder, thereby bypassing any user knowledge of the events. This approach requires certain DVD-Video authoring requirements: (1) interleaving video and audio streams to prevent "back door" playback/access; (2) ability to populate GPRMs to create locking sequence.

There are two parts to the DVD-Video unlocking mechanism:

- The actual unlock process – performs actual unlocking of video. Without unlock process, consumer cannot access video. Therefore software on the remote server is required to unlock the video and other players will not support this feature.
- The protection process – the process protects against malicious consumers who try to bypass unlocking process. This process is not an actual locking process, but manipulates and distorts the video, thus rendering it non-viewable by consumers that try to bypass the unlock mechanism.

Unlock process:

The locking process is performed during the video authoring process of the DVD-Video. Each portion of video to be unlocked can be authored into a separate title, or title/chapter combination. (All references to locking video also apply to locking DVD-Audio)

The locking of the video utilizes General Parameter Registers (GPRMs), which are inherent in the DVD-Video specification. A DVD-Video title can be authored such that the GPRMs must be set to a specific value in order to allow an action to occur. In the case of our video unlocking scheme, the process of locking a video is ensuring that a video can only be played when the GPRMs are properly set. Then the DVD-Video is authored in such a way that remote server operator can programmatically (without consumer interaction) set the value of the GPRMs when certain conditions/transaction criteria has been met.

Fig. 26 illustrates a method 2602 for unlocking content. The steps required to create the "lock" or "key" for unlocking begin with a step 2604 of creating a title, the title being some form of content such as a movie video, training video or some other form of content. Thereafter, in a step 2606 hot spots are authored in the title (or use post command and jump to menu that contains hot spots). Then, in a step

2608, all of the hotspots are overlapped. In a step 2610 a sequence of events that trigger GPRMs is passed to the client. Once correct sequence of events are passed to client, resulting in populating the GPRMs properly, then, in a step 2612 the consumer is allowed to play the locked video.

Once correct GPRMs are populated, the appropriate post command is generated to jump the consumer to the locked title. Also note the GPRMs can be populated by either a direct call to an interface that allows setting the GPRM bits or through menu navigation commands such as left, right, up, and down.

It is also recommended to create a TIFF or animated graphic that displays when the DVD is placed into a traditional DVD consumer player. This TIFF or animated graphic can inform the user to place the DVD into a computer to access the special features and unlock the appropriate content. This information is displayed as soon as the First Play PGC is encountered.

Protection Process:

To avoid DVD playback solutions that violate the DVD-Video guidelines, additional precautions should be taken (these are not required, but recommended):

- Each section of video (title or chapter) to be locked should be interleaved with another portion of video (such as black, or with a second video to be locked. Since this requires the use of multi-angle content, the two sections of video must be of the exact length. This avoids certain decoders which ignore the DVD-Video layout and allow the consumer to play a VOB file directly. The first/default video stream (angle 1) should be the black, dummy video. The second video stream (angle 2) should be the actual content to be unlocked.
- Each section of video to be locked should contain two audio tracks: the first/default audio track should be noise, garbage or any audio that is NOT associated with the content to be unlocked. This is the default audio track that will be played if the consumer attempts to bypass the DVD-Video specification. The second audio track should be the actual audio track for the locked video.

This protection process is useful because interlaced/multi-angle video is formatted out sequentially. In other words, interlaced content is stored in the following manner: Seconds 0-2 video 1, seconds 0-2 video 2, seconds 3-5 video 1, seconds 3-5 video 2, etc.

Therefore if a consumer attempts to play the underlying VOB file directly, they will see video alternating every two seconds – which is very annoying. Additionally, if a VOB file is played using this approach, the default audio stream will play, which as defined above, will be garbage.

If utilizing these protection processes: after unlock process has completed successfully, the DVD-Video should be authored to play the appropriate title/chapter combination, as well as defaulting to the second video stream (angle 2) and second audio stream (audio track 2). Relate to BCA based on distribution channel allow access to content.

Control is on the Server side and it controls the client. Therefore server can give commands for content. The server can also create a game out of the navigation if they view certain clips in a certain order they will essentially be walking through a key setting scheme and thus unlocking further content. The Unlock information on a web site by requiring a DVD to be in the drive. The BCA number or Disk ID is passed to the website in the HTTP header and then this allows the content on the website to be accessed.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any other the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

CLAIMS

What is claimed is:

1. A method for synchronizing an event on a plurality of a client apparatuses, comprising the steps of:

(a) providing an event stored in a memory storage device;

(b) connecting the client apparatuses and a host computer to a network; and

(c) transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

2. A method as recited in claim 1, wherein the event includes a video and audio presentation.

3. A method as recited in claim 2, wherein the event includes at least one of a movie, a concert, and a theatrical event.

4. A method as recited in claim 1, wherein the network is a wide area network.

5. A method as recited in claim 1, wherein the information includes a start time when the playback of the event is to begin on each of the client apparatuses.

6. A method as recited in claim 1, wherein the information includes an ending time when the playback of the event is to end on each of the client apparatuses.

7. A method as recited in claim 1, wherein the memory storage device includes a digital video disc (DVD) player.

8. A method as recited in claim 7, wherein the information includes chapter information associated with the DVD.

9. A method as recited in claim 1, and further comprising the step of receiving input from the user, and altering the playback based on the input.

10. A computer program embodied on a computer readable medium for synchronizing an event on a plurality of a client apparatuses, comprising:

(a) a code segment for providing an event stored in a memory storage device;

(b) a code segment for connecting the client apparatuses and a host computer to a network; and

(c) a code segment for transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

11. A computer program as recited in claim 10, wherein the event includes a video and audio presentation.

12. A computer program as recited in claim 11, wherein the event includes at least one of a movie, a concert, and a theatrical event.

13. A computer program as recited in claim 10, wherein the network is a wide area network.

14. A computer program as recited in claim 10, wherein the information includes a start time when the playback of the event is to begin on each of the client apparatuses.

15. A computer program as recited in claim 10, wherein the information includes an ending time when the playback of the event is to end on each of the client apparatuses.

16. A computer program as recited in claim 10, wherein the memory storage device includes a digital video disc (DVD) player.

17. A computer program as recited in claim 16, wherein the information includes chapter information associated with the DVD.

18. A computer program as recited in claim 10, and further comprising a code segment for receiving input from the user, and altering the playback based on the input.

19. A system for synchronizing an event on a plurality of a client apparatuses, comprising:

(a) logic for providing an event stored in a memory storage device;

(b) logic for connecting the client apparatuses and a host computer to a network; and

(c) logic for transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

20. A method for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising the steps of:

(a) providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

21. A method as recited in claim 20, wherein the event includes a video and audio presentation.

22. A method as recited in claim 20, wherein the information includes a history and data associated with the simultaneous playback.

23. A method as recited in claim 20, wherein the network is a wide area network.

24. A method as recited in claim 20, wherein the memory includes a digital video disc (DVD).

25. A method as recited in claim 24, wherein the information includes chapter information associated with the DVD.

26. A computer program embodied on a computer readable medium for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising:

(a) a code segment for providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) a code segment for storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) a code segment for allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

27. A computer program as recited in claim 26, wherein the event includes a video and audio presentation.

28. A computer program as recited in claim 26, wherein the information includes a history and data associated with the simultaneous playback.

29. A computer program as recited in claim 26, wherein the network is a wide area network.

30. A computer program as recited in claim 26, wherein the memory includes a digital video disc (DVD).

31. A computer program as recited in claim 26, wherein the information includes chapter information associated with the DVD.

32. A system for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising:

(a) logic for providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) logic for storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) logic for allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

33. A system as recited in claim 32, wherein the event includes a video and audio presentation.

34. A system as recited in claim 32, wherein the information includes a history and data associated with the simultaneous playback.

35. A system as recited in claim 32, wherein the network is a wide area network.

36. A system as recited in claim 32, wherein the memory includes a digital video disc (DVD).

37. A system as recited in claim 32, wherein the information includes chapter information associated with the DVD.

38. A method for providing overlays during a synchronized event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network;

(b) simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) overlaying material during the playback of the event based on input received utilizing the network.

39. A method as recited in claim 38, wherein the overlay material includes audible material.

40. A method as recited in claim 38, wherein the overlay material includes visible material.

41. A method as recited in claim 40, wherein the overlay material includes annotations on a display of the client apparatus.

42. A method as recited in claim 40, and further comprising the step of displaying the overlay material on each of the client apparatuses utilizing the network.

43. A method as recited in claim 42, wherein the overlay material is displayed during the simultaneous playback of the event.

44. A method as recited in claim 42, and further comprising the step of identifying the client apparatus that provided the overlay material.

45. A method as recited in claim 38, wherein the event is stored on a digital video disc (DVD).

46. A method as recited in claim 38, and further comprising the step of receiving the input, and altering the playback based on the input.

47. A computer program embodied on a computer readable medium for providing overlays during a synchronized event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network;

(b) a code segment for simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) a code segment for overlaying material during the playback of the event based on input received utilizing the network.

48. A computer program as recited in claim 47, wherein the overlay material includes audible material.

49. A computer program as recited in claim 47, wherein the overlay material includes visible material.

50. A computer program as recited in claim 49, wherein the overlay material includes annotations on a display of the client apparatus.

51. A computer program as recited in claim 49, and further comprising a code segment for displaying the overlay material on each of the client apparatuses utilizing the network.

52. A computer program as recited in claim 51, wherein the overlay material is displayed during the simultaneous playback of the event.

53. A computer program as recited in claim 51, and further comprising the a code segment for identifying the client apparatus that provided the overlay material.

54. A computer program as recited in claim 47, wherein the event is stored on a digital video disc (DVD).

55. A computer program as recited in claim 47, and further comprising a code segment for receiving the input, and altering the playback based on the input.

56. A system for providing overlays during a synchronized event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network;

(b) logic for simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) logic for overlaying material during the playback of the event based on input received utilizing the network.

57. A method for delayed synchronization of an event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) simultaneously playing back the event on the client apparatuses utilizing the network;

(c) receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

58. A method as recited in claim 57, wherein the request is received utilizing the network.

59. A method as recited in claim 57, wherein the event includes at least one of a movie, a concert, and a theatrical event.

60. A method as recited in claim 57, wherein the memory includes a digital video disc (DVD).

61. A method as recited in claim 60, wherein the information includes chapter information associated with the DVD.

62. A method as recited in claim 57, wherein the network is a wide area network.

63. A computer program embodied on a computer readable medium for delayed synchronization of an event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) a code segment for simultaneously playing back the event on the client apparatuses utilizing the network;

(c) a code segment for receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) a code segment for transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

64. A computer program as recited in claim 63, wherein the request is received utilizing the network.

65. A computer program as recited in claim 63, wherein the event includes at least one of a movie, a concert, and a theatrical event.

66. A computer program as recited in claim 63, wherein the memory includes a digital video disc (DVD).

67. A computer program as recited in claim 66, wherein the information includes chapter information associated with the DVD.

68. A computer program as recited in claim 63, wherein the network is a wide area network.

69. A system for delayed synchronization of an event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) logic for simultaneously playing back the event on the client apparatuses utilizing the network;

(c) logic for receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) logic for transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

70. A system as recited in claim 69, wherein the request is received utilizing the network.

71. A system as recited in claim 69, wherein the event includes at least one of a movie, a concert, and a theatrical event.

72. A system as recited in claim 69, wherein the memory includes a digital video disc (DVD).

73. A system as recited in claim 72, wherein the information includes chapter information associated with the DVD.

74. A system as recited in claim 69, wherein the network is a wide area network.

75. A method for synchronizing an event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network;

(b) embedding an application program on a site on the network;

(c) requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) receiving a script for displaying the information.

76. A method as recited in claim 75, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

77. A method as recited in claim 76, wherein the playback device includes a digital video disc (DVD) player.

78. A method as recited in claim 76, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

79. A method as recited in claim 76, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

80. A method as recited in claim 75, wherein application program is a JAVA applet and the script is JAVAscript.

81. A computer program embodied on a computer readable medium for synchronizing an event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network;

(b) a code segment for embedding an application program on a site on the network;

(c) a code segment for requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) a code segment for receiving a script for displaying the information.

82. A computer program as recited in claim 81, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

83. A computer program as recited in claim 82, wherein the playback device includes a digital video disc (DVD) player.

84. A computer program as recited in claim 82, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

85. A computer program as recited in claim 82, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

86. A computer program as recited in claim 81, wherein application program is a JAVA applet and the script is JAVAscript.

87. A system for synchronizing an event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network;

(b) logic for embedding an application program on a site on the network;

(c) logic for requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) logic for receiving a script for displaying the information.

88. A system as recited in claim 87, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

89. A system as recited in claim 88, wherein the playback device includes a digital video disc (DVD) player.

90. A system as recited in claim 88, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

91. A system as recited in claim 88, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

92. A system as recited in claim 87, wherein application program is a JAVA applet and the script is JAVAscript.

93. A method for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising the steps of:

(a) receiving a request utilizing a network for viewing an event;

(b) queuing the request in memory;

(c) creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) sending the object to one of the client apparatuses utilizing the network for being stored therein.

94. A method as recited in claim 93, wherein the request is received via an application program embedded in a site on the network.

95. A method as recited in claim 94, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

96. A method as recited in claim 95, wherein the memory includes a digital video disc (DVD).

97. A method as recited in claim 93, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

98. A method as recited in claim 93, wherein the activation signal is provided using a clock of the client apparatus.

99. A computer program embodied on a computer readable medium for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising:

(a) a code segment for receiving a request utilizing a network for viewing an event;

(b) a code segment for queuing the request in memory;

(c) a code segment for creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) a code segment for sending the object to one of the client apparatuses utilizing the network for being stored therein.

100. A computer program as recited in claim 99, wherein the request is received via an application program embedded in a site on the network.

101. A computer program as recited in claim 100, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

102. A computer program as recited in claim 101, wherein the memory includes a digital video disc (DVD).

103. A computer program as recited in claim 99, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

104. A computer program as recited in claim 99, wherein the activation signal is provided using a clock of the client apparatus.

105. A system for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising:

(a) logic for receiving a request utilizing a network for viewing an event;

(b) logic for queuing the request in memory;

(c) logic for creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) logic for sending the object to one of the client apparatuses utilizing the network for being stored therein.

106. A system as recited in claim 105, wherein the request is received via an application program embedded in a site on the network.

107. A system as recited in claim 106, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

108. A system as recited in claim 107, wherein the memory includes a digital video disc (DVD).

109. A system as recited in claim 105, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

110. A system as recited in claim 105, wherein the activation signal is provided using a clock of the client apparatus.

111. A method for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising the steps of:

(a) determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) calculating a length of the event based on the start time and the stop time;

(c) storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) creating a loop at the start time during which a lapsed time of the event is tracked.

112. A method as recited in claim 111, wherein the current time is determined by querying a clock of one of the client apparatuses.

113. A method as recited in claim 111, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

114. A method as recited in claim 111, and further comprising the step of storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

115. A method as recited in claim 111, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

116. A method as recited in claim 115, and further comprising the step of creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

117. A computer program embodied on a computer readable medium for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising:

(a) a code segment for determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) a code segment for calculating a length of the event based on the start time and the stop time;

(c) a code segment for storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) a code segment for creating a loop at the start time during which a lapsed time of the event is tracked.

118. A computer program as recited in claim 117, wherein the current time is determined by querying a clock of one of the client apparatuses.

119. A computer program as recited in claim 117, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

120. A computer program as recited in claim 117, and further comprising a code segment for storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

121. A computer program as recited in claim 117, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

122. A computer program as recited in claim 115, and further comprising a code segment for creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

123. A system for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising:

(a) logic for determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) logic for calculating a length of the event based on the start time and the stop time;

(c) logic for storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) logic for creating a loop at the start time during which a lapsed time of the event is tracked.

124. A system as recited in claim 123, wherein the current time is determined by querying a clock of one of the client apparatuses.

125. A system as recited in claim 123, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

126. A system as recited in claim 123, and further comprising logic for storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

127. A system as recited in claim 123, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

128. A system as recited in claim 127, and further comprising logic for creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

129. A method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising the steps of:

(a) providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) comparing the identifier with an identifier of a scheduled event; and

(d) beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

130. A method as recited in claim 129, wherein the event includes a video and audio presentation.

131. A method as recited in claim 129, wherein the event includes at least one of a movie, a concert, and a theatrical event.

132. A method as recited in claim 129, wherein the network is a wide area network.

133. A method as recited in claim 129, wherein the memory includes a digital video disc (DVD).

134. A computer program embodied on a computer readable medium for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising:

(a) a code segment for providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) a code segment for ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) a code segment for comparing the identifier with an identifier of a scheduled event; and

(d) a code segment for beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

135. A computer program as recited in claim 134, wherein the event includes a video and audio presentation.

136. A computer program as recited in claim 134, wherein the event includes at least one of a movie, a concert, and a theatrical event.

137. A computer program as recited in claim 134, wherein the network is a wide area network.

138. A computer program as recited in claim 134, wherein the memory includes a digital video disc (DVD).

139. A system for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising:

(a) logic for providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) logic for ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) logic for comparing the identifier with an identifier of a scheduled event; and

(d) logic for beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

140. A system as recited in claim 139, wherein the event includes a video and audio presentation.

141. A system as recited in claim 139, wherein the event includes at least one of a movie, a concert, and a theatrical event.

142. A system as recited in claim 139, wherein the network is a wide area network.

143. A system as recited in claim 139, wherein the memory includes a digital video disc (DVD)..

144. A method for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising the steps of:

(a) identifying a type of the playback device of each of the client apparatuses;

(b) looking up a command associated with the identified type of the playback device; and

(c) sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

145. A method as recited in claim 144, wherein the event includes a video and audio presentation.

146. A method as recited in claim 144, wherein the type of the playback device is identified utilizing the network.

147. A method as recited in claim 144, wherein the network is a wide area network.

148. A method as recited in claim 144, and further comprising the step of storing on the client apparatus an identifier of a host server that sent the command.

149. A method as recited in claim 144, wherein the memory includes a digital video disc (DVD).

150. A computer program embodied on a computer readable medium for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising:

(a) a code segment for identifying a type of the playback device of each of the client apparatuses;

(b) a code segment for looking up a command associated with the identified type of the playback device; and

(c) a code segment for sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

151. A computer program as recited in claim 150, wherein the event includes a video and audio presentation.

152. A computer program as recited in claim 150, wherein the type of the playback device is identified utilizing the network.

153. A computer program as recited in claim 150, wherein the network is a wide area network.

154. A computer program as recited in claim 150, and further comprising a code segment for storing on the client apparatus an identifier of a host server that sent the command.

155. A computer program as recited in claim 150, wherein the memory includes a digital video disc (DVD).

156. A system for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising:

(a) logic for identifying a type of the playback device of each of the client apparatuses;

(b) logic for looking up a command associated with the identified type of the playback device; and

(c) logic for sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

157. A system as recited in claim 156, wherein the event includes a video and audio presentation.

158. A system as recited in claim 156, wherein the type of the playback device is identified utilizing the network.

159. A system as recited in claim 156, wherein the network is a wide area network.

160. A system as recited in claim 156, and further comprising logic for storing on the client apparatus an identifier of a host server that sent the command.

161. A system as recited in claim 156, wherein the memory includes a digital video disc (DVD).

162. A method for remotely controlling digital content, the digital content being stored locally on a client device, the method comprising the steps of:

(a) coupling the client with a network to retrieve a client identification from the client device;

(b) generating a query, based upon the client identification, to determine whether the client should have access to the digital content; and

(c) sending an unlock command via the network to the client device, if the client should have access to the content stored on the client device, the command being operable to allow the client device to access and utilize the digital content stored locally on the client device.

163. A method for remotely controlling content as recited in claim 162 wherein the client identification includes an identification of a user operating the client device.

164. A method for remotely controlling digital content as recited in claim 162 wherein the client identification includes an identification of the client device.

165. A method for remotely controlling digital content as recited in claim 162, including the step of sending a navigation command via the network to the client device, the navigation command being operable to control navigation of the content stored locally on the client device.

166. A method for remotely controlling content as recited in claim 165, including the step of delivering additional content to the client device via the network, to supplement the content stored locally thereon.

167. A method for remotely controlling digital content as recited in claim 162 wherein the client device is a computer.

168. A method for remotely controlling digital content as recited in claim 162 wherein the client device is a set top box.

169. A method for remotely controlling digital content as recited in claim 162 wherein the locally stored content is embodied on a Digital Versatile Disc (DVD).

170. A method for remotely controlling content as recited in claim 162, including the step of initiating synchronous play of content on a plurality of client devices, each client device being connected to the network.

171. A method for remotely controlling digital content as recited in claim 162 wherein the network is the Internet.

172. A method for remotely controlling digital content as recited in claim 162 wherein the unlock command incrementally allows access to portions of the content stored on the client device based upon the determination of whether the client should have access to the content.

173. A computer program embodied on a computer readable medium for remotely controlling digital content which has been stored locally on a client device, the computer program comprising:

(a) a code segment that receives an input delivered over a network from the client device, the input including a client identifier;

(b) a code segment that queries the client input to determine, based upon the client identifier, whether the client should have access to the content;

(c) a code segment that unlocks the digital content stored on locally on the client device, allowing access to the content by the client device based upon the results of the query; and

(d) a code segment that delivers to the client device, via the network, the code segment that unlocks the locally stored content.

174. A computer program as recited in claim 173 wherein the client identifier includes an identification of the client device.

175. A computer program as recited in claim 173 wherein the client identifier includes an identification of a user operating the client device.

176. A computer program as recited in claim 175, including a code segment for supplying additional content to the client device to supplement the content stored locally thereon based upon the client identifier of the client device.

177. A computer program as recited in claim 173 wherein the digital content is a video stored on a Digital Versatile Disc (DVD).

178. A computer program as recited in claim 173 wherein the code segment for controlling the use of the content controls navigation of the content by the client device.

179. A computer program as recited in claim 178 further comprising a code segment for synchronizing play of the video on a plurality of client devices.

180. A system for remotely controlling digital content, the digital content being stored locally on a client device, the system comprising:

(a) a processor, remote from the client device;

(b) a memory, remote from the client device, that stores information under control of the processor;

(c) logic stored on the memory that retrieves and interprets input from the client device, the input being delivered over a network and including a client device identification;

(d) logic stored on the memory, that responds to the input from the client device to generate an unlock command, based upon the client device identification, to allow use of the locally stored digital content by the client device having the digital content stored thereon; and

(e) logic stored on the memory that delivers the unlock command over the network to the client device.

181. A system as recited in claim 180 wherein the client identification includes an identification of a user operating the client device.

182. A system as recited in claim 180 wherein the client identification includes an identification of the client device;

183. A system as recited in claim 180 including logic that controls navigation of the digital content stored locally on the client device.

184. A system as recited in claim 180, including logic that supplements the content stored locally on the client device with additional content delivered over the network.

185. A system as recited in claim 180 wherein the digital content on client device is initially embodied on a Digital Versatile Disk (DVD).

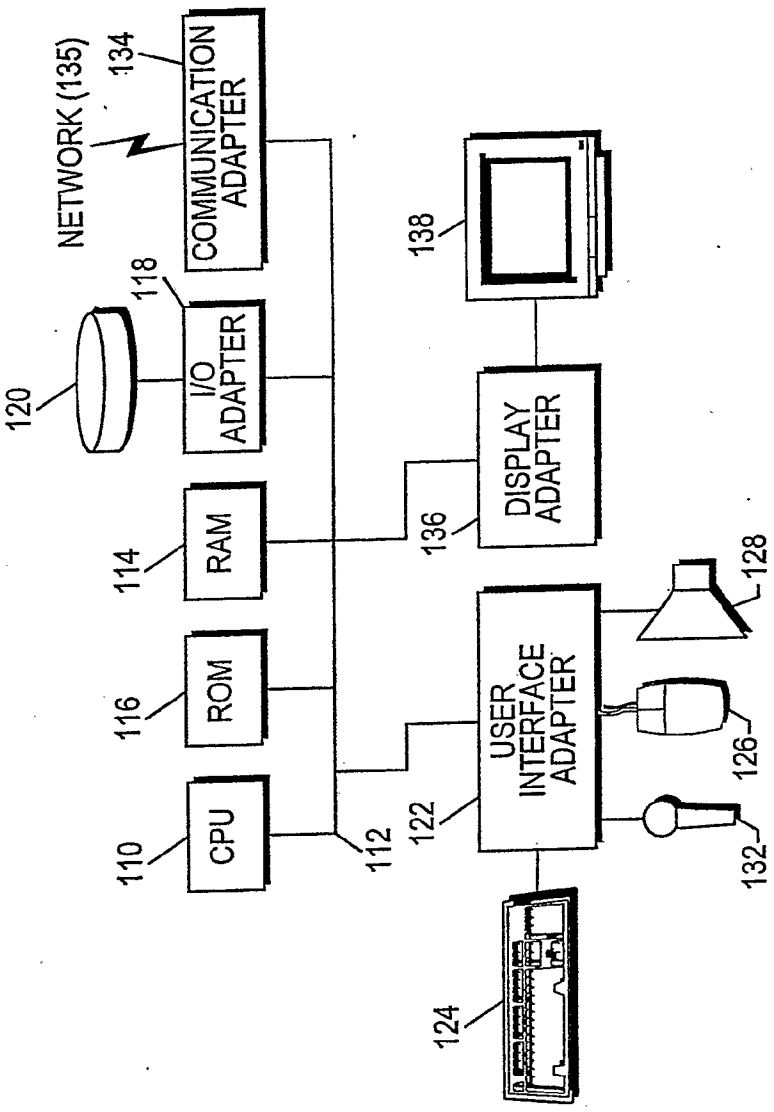

NETWORK (135)
134
COMMUNICATION ADAPTER
118
I/O ADAPTER
120
114
RAM
116
ROM
110
CPU
112
122
USER INTERFACE ADAPTER
136
DISPLAY ADAPTER
138
128
126
132
124

Figure 1

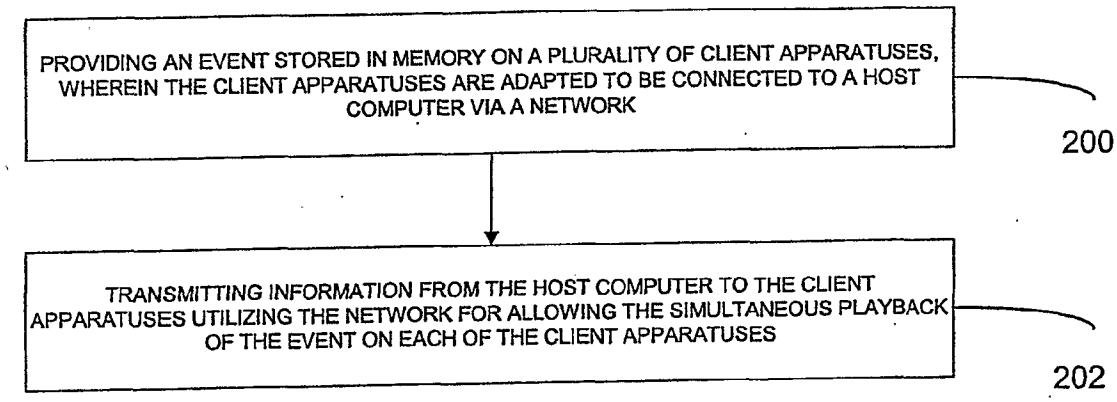

PROVIDING AN EVENT STORED IN MEMORY ON A PLURALITY OF CLIENT APPARATUSES, WHEREIN THE CLIENT APPARATUSES ARE ADAPTED TO BE CONNECTED TO A HOST COMPUTER VIA A NETWORK
200
TRANSMITTING INFORMATION FROM THE HOST COMPUTER TO THE CLIENT APPARATUSES UTILIZING THE NETWORK FOR ALLOWING THE SIMULTANEOUS PLAYBACK OF THE EVENT ON EACH OF THE CLIENT APPARATUSES
202

Figure 2

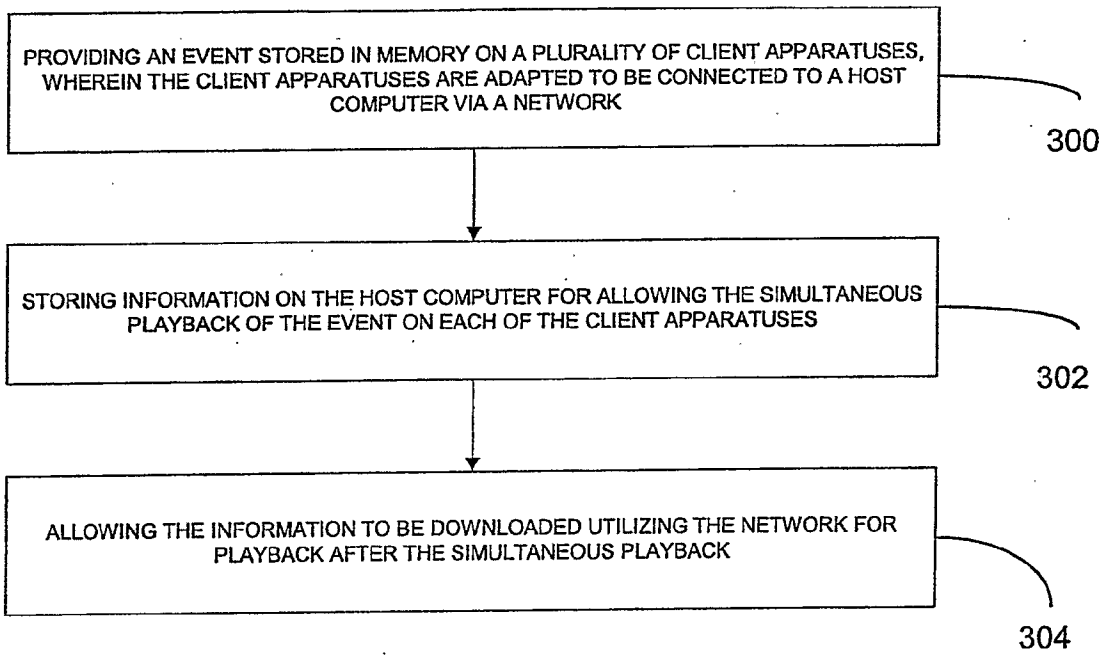

PROVIDING AN EVENT STORED IN MEMORY ON A PLURALITY OF CLIENT APPARATUSES, WHEREIN THE CLIENT APPARATUSES ARE ADAPTED TO BE CONNECTED TO A HOST COMPUTER VIA A NETWORK
300
STORING INFORMATION ON THE HOST COMPUTER FOR ALLOWING THE SIMULTANEOUS PLAYBACK OF THE EVENT ON EACH OF THE CLIENT APPARATUSES
302
ALLOWING THE INFORMATION TO BE DOWNLOADED UTILIZING THE NETWORK FOR PLAYBACK AFTER THE SIMULTANEOUS PLAYBACK
304

Figure 3

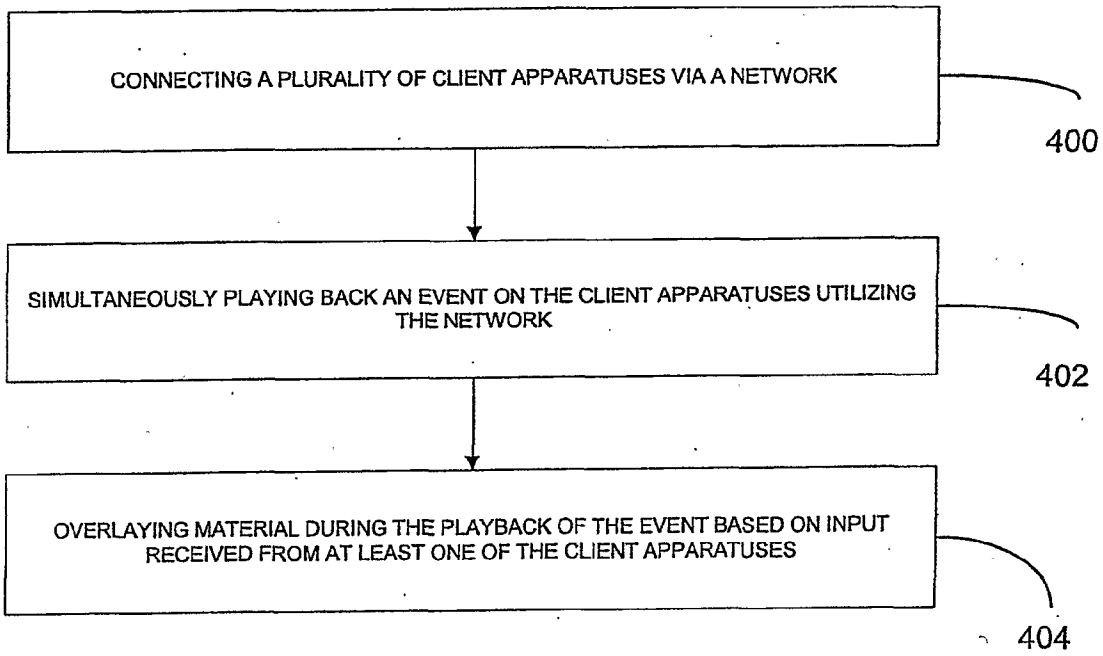

CONNECTING A PLURALITY OF CLIENT APPARATUSES VIA A NETWORK
400
SIMULTANEOUSLY PLAYING BACK AN EVENT ON THE CLIENT APPARATUSES UTILIZING THE NETWORK
402
OVERLAYING MATERIAL DURING THE PLAYBACK OF THE EVENT BASED ON INPUT RECEIVED FROM AT LEAST ONE OF THE CLIENT APPARATUSES
404

Figure 4

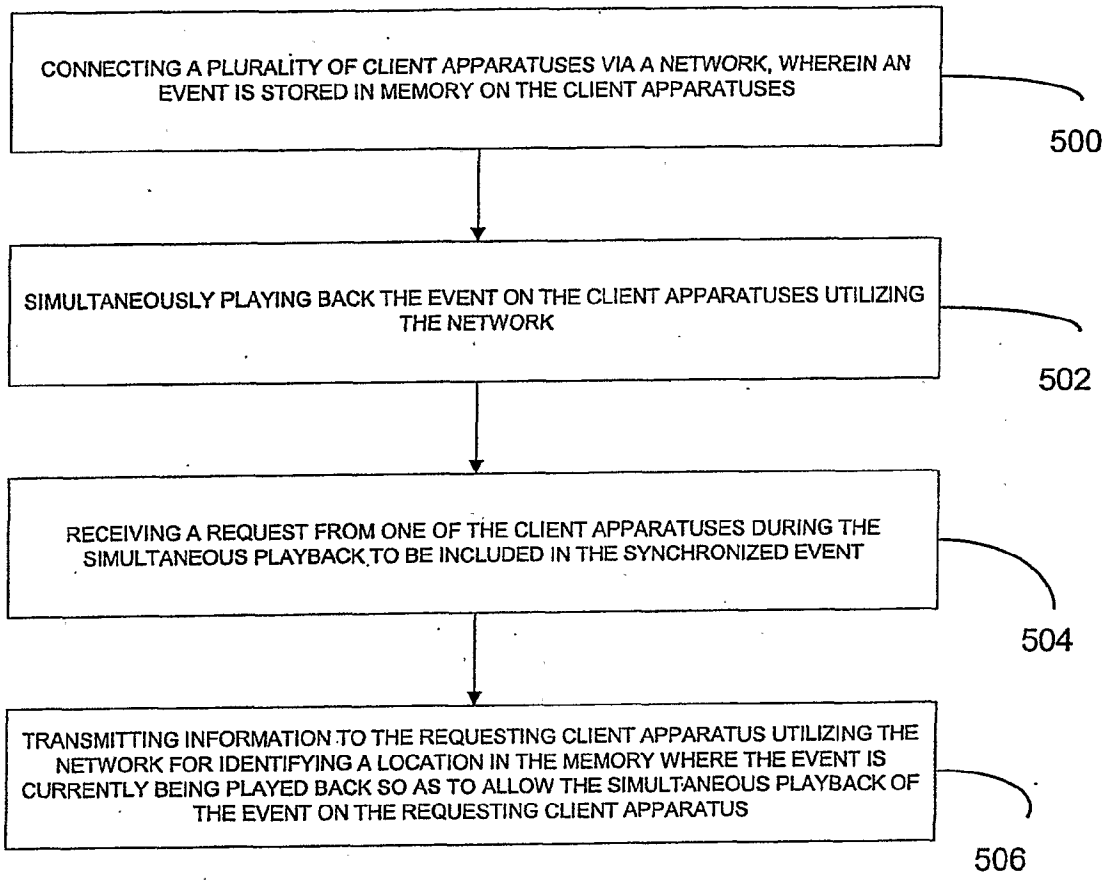

CONNECTING A PLURALITY OF CLIENT APPARATUSES VIA A NETWORK, WHEREIN AN EVENT IS STORED IN MEMORY ON THE CLIENT APPARATUSES
500
SIMULTANEOUSLY PLAYING BACK THE EVENT ON THE CLIENT APPARATUSES UTILIZING THE NETWORK
502
RECEIVING A REQUEST FROM ONE OF THE CLIENT APPARATUSES DURING THE SIMULTANEOUS PLAYBACK TO BE INCLUDED IN THE SYNCHRONIZED EVENT
504
TRANSMITTING INFORMATION TO THE REQUESTING CLIENT APPARATUS UTILIZING THE NETWORK FOR IDENTIFYING A LOCATION IN THE MEMORY WHERE THE EVENT IS CURRENTLY BEING PLAYED BACK SO AS TO ALLOW THE SIMULTANEOUS PLAYBACK OF THE EVENT ON THE REQUESTING CLIENT APPARATUS
506

Figure 5

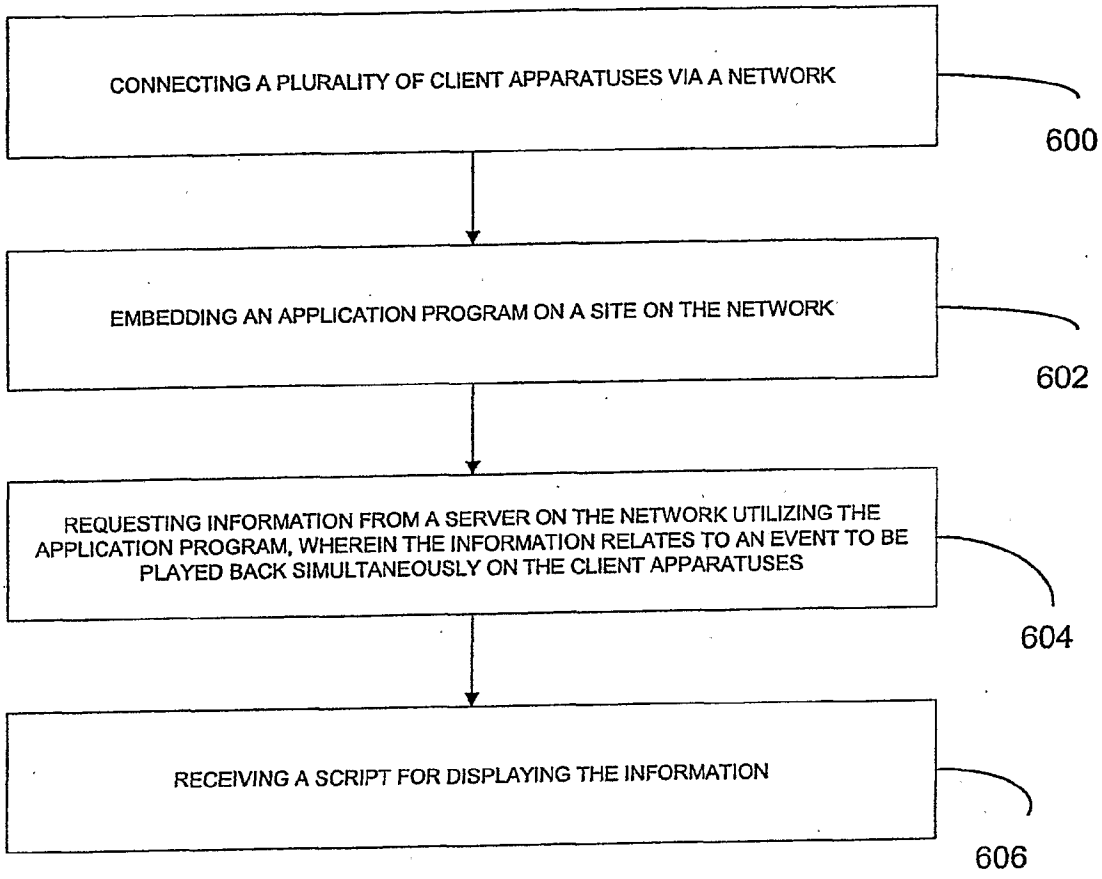

CONNECTING A PLURALITY OF CLIENT APPARATUSES VIA A NETWORK
600
EMBEDDING AN APPLICATION PROGRAM ON A SITE ON THE NETWORK
602
REQUESTING INFORMATION FROM A SERVER ON THE NETWORK UTILIZING THE APPLICATION PROGRAM, WHEREIN THE INFORMATION RELATES TO AN EVENT TO BE PLAYED BACK SIMULTANEOUSLY ON THE CLIENT APPARATUSES
604
RECEIVING A SCRIPT FOR DISPLAYING THE INFORMATION
606

Figure 6

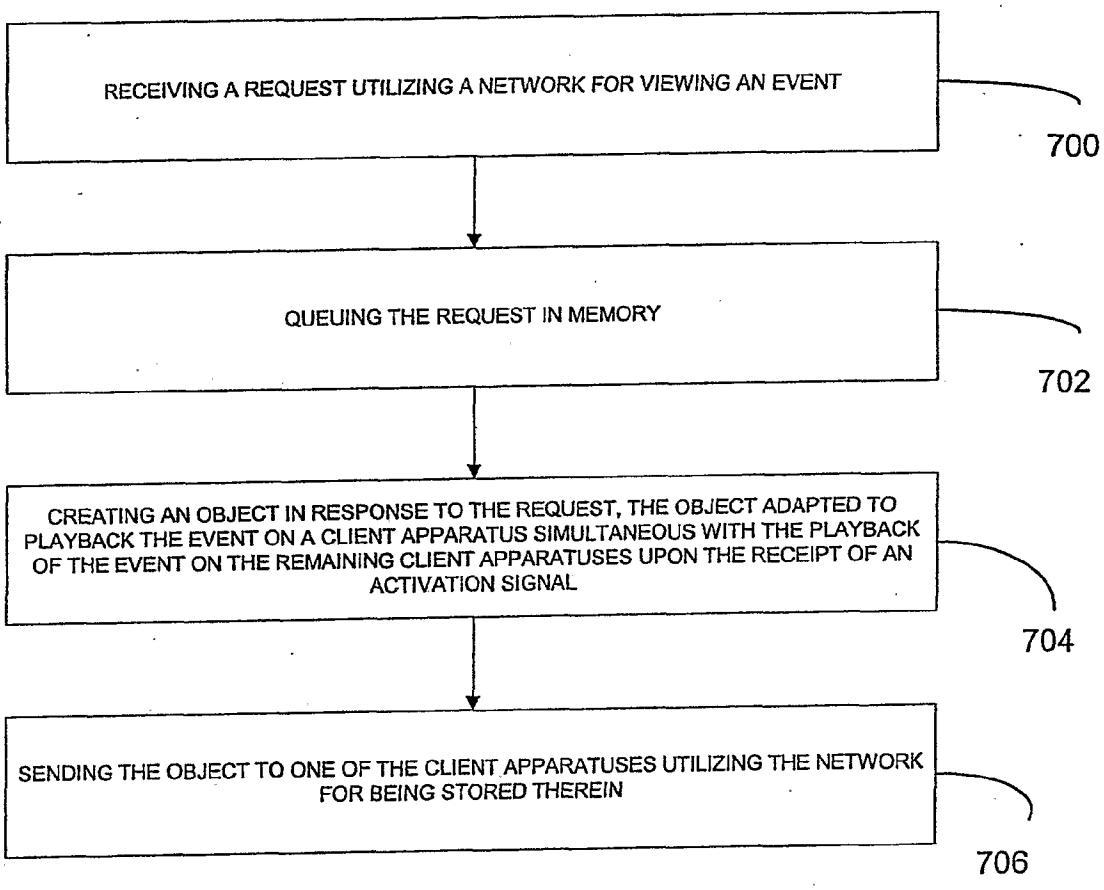

RECEIVING A REQUEST UTILIZING A NETWORK FOR VIEWING AN EVENT
700
QUEUING THE REQUEST IN MEMORY
702
CREATING AN OBJECT IN RESPONSE TO THE REQUEST, THE OBJECT ADAPTED TO PLAYBACK THE EVENT ON A CLIENT APPARATUS SIMULTANEOUS WITH THE PLAYBACK OF THE EVENT ON THE REMAINING CLIENT APPARATUSES UPON THE RECEIPT OF AN ACTIVATION SIGNAL
704
SENDING THE OBJECT TO ONE OF THE CLIENT APPARATUSES UTILIZING THE NETWORK FOR BEING STORED THEREIN
706

Figure 7

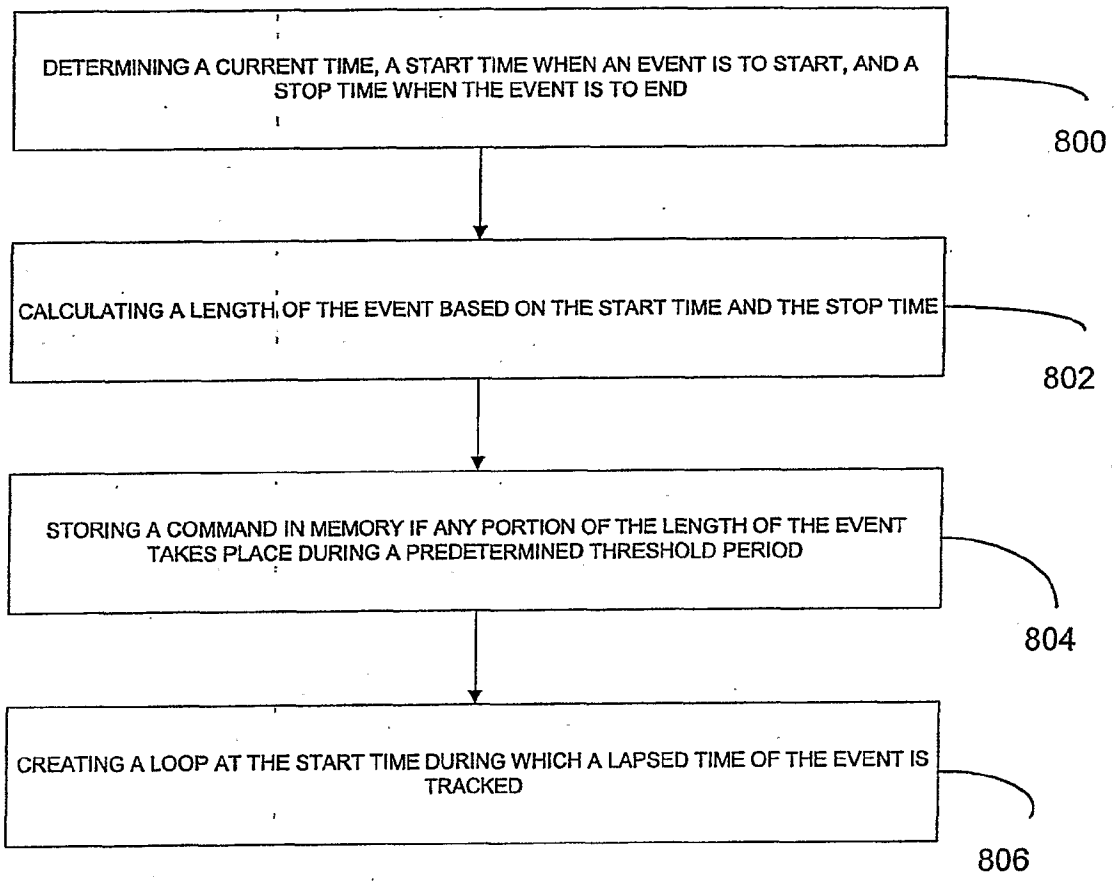

DETERMINING A CURRENT TIME, A START TIME WHEN AN EVENT IS TO START, AND A STOP TIME WHEN THE EVENT IS TO END
800
CALCULATING A LENGTH OF THE EVENT BASED ON THE START TIME AND THE STOP TIME
802
STORING A COMMAND IN MEMORY IF ANY PORTION OF THE LENGTH OF THE EVENT TAKES PLACE DURING A PREDETERMINED THRESHOLD PERIOD
804
CREATING A LOOP AT THE START TIME DURING WHICH A LAPSED TIME OF THE EVENT IS TRACKED
806

Figure 8

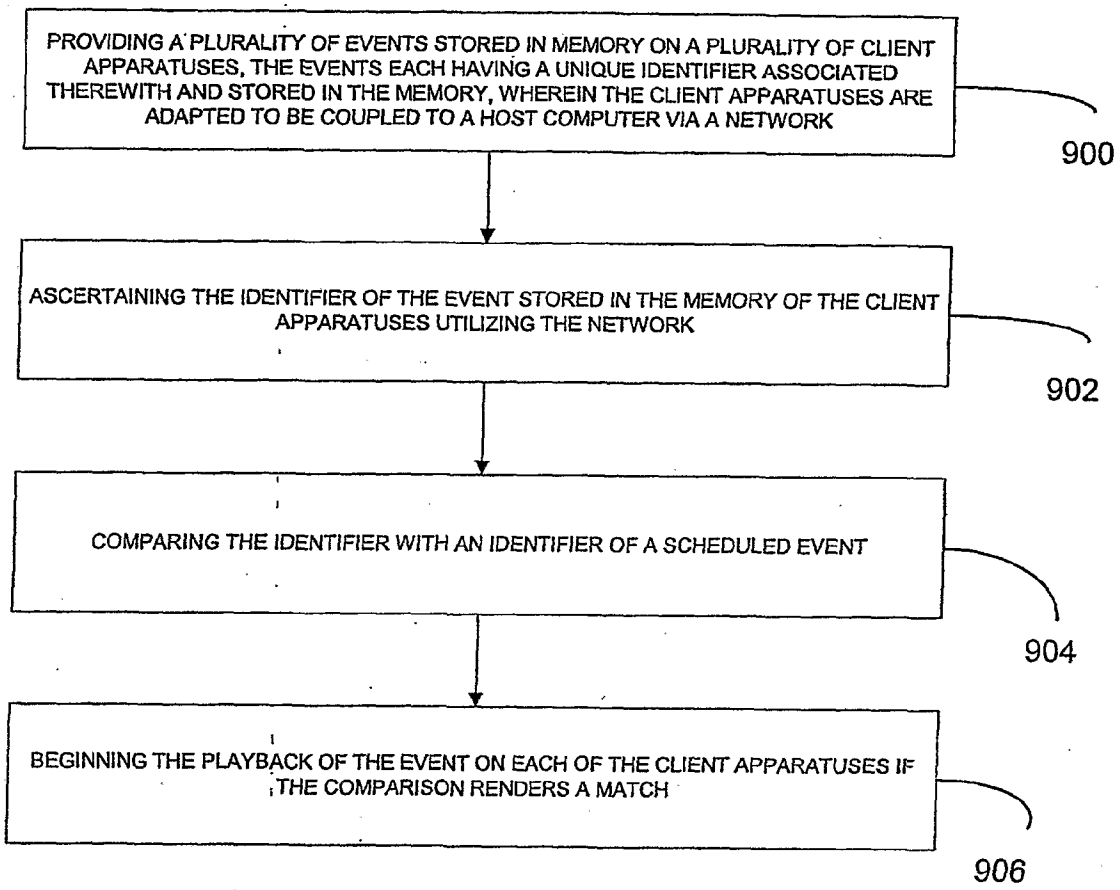

PROVIDING A PLURALITY OF EVENTS STORED IN MEMORY ON A PLURALITY OF CLIENT APPARATUSES, THE EVENTS EACH HAVING A UNIQUE IDENTIFIER ASSOCIATED THEREWITH AND STORED IN THE MEMORY, WHEREIN THE CLIENT APPARATUSES ARE ADAPTED TO BE COUPLED TO A HOST COMPUTER VIA A NETWORK
900
ASCERTAINING THE IDENTIFIER OF THE EVENT STORED IN THE MEMORY OF THE CLIENT APPARATUSES UTILIZING THE NETWORK
902
COMPARING THE IDENTIFIER WITH AN IDENTIFIER OF A SCHEDULED EVENT
904
BEGINNING THE PLAYBACK OF THE EVENT ON EACH OF THE CLIENT APPARATUSES IF THE COMPARISON RENDERS A MATCH
906

Figure 9

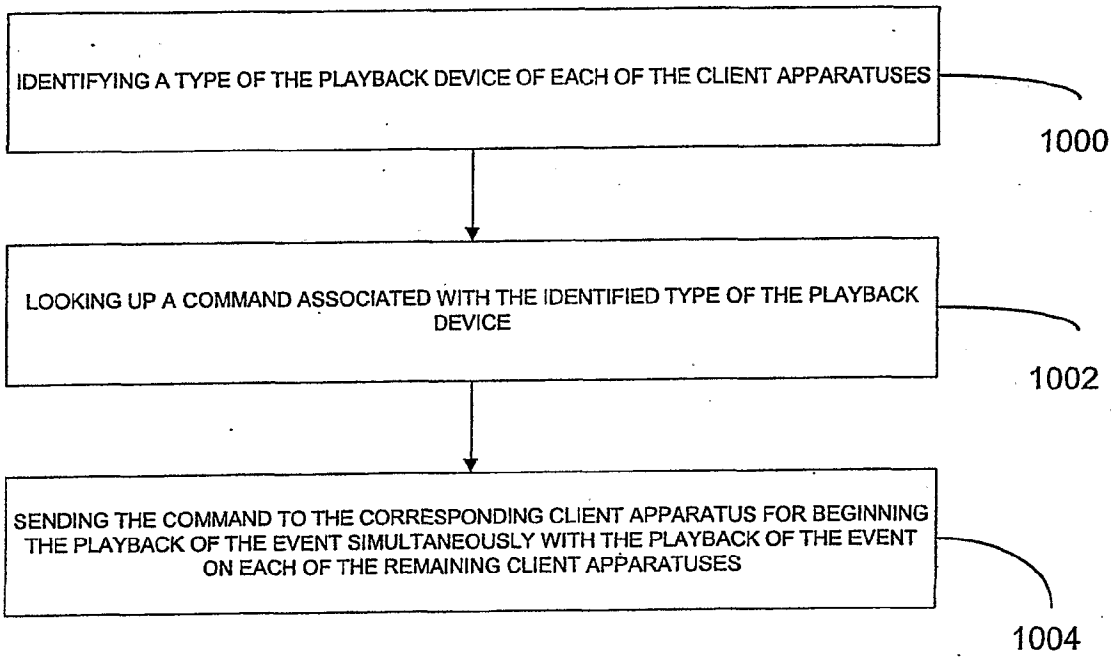

IDENTIFYING A TYPE OF THE PLAYBACK DEVICE OF EACH OF THE CLIENT APPARATUSES
1000
LOOKING UP A COMMAND ASSOCIATED WITH THE IDENTIFIED TYPE OF THE PLAYBACK DEVICE
1002
SENDING THE COMMAND TO THE CORRESPONDING CLIENT APPARATUS FOR BEGINNING THE PLAYBACK OF THE EVENT SIMULTANEOUSLY WITH THE PLAYBACK OF THE EVENT ON EACH OF THE REMAINING CLIENT APPARATUSES
1004

Figure 10

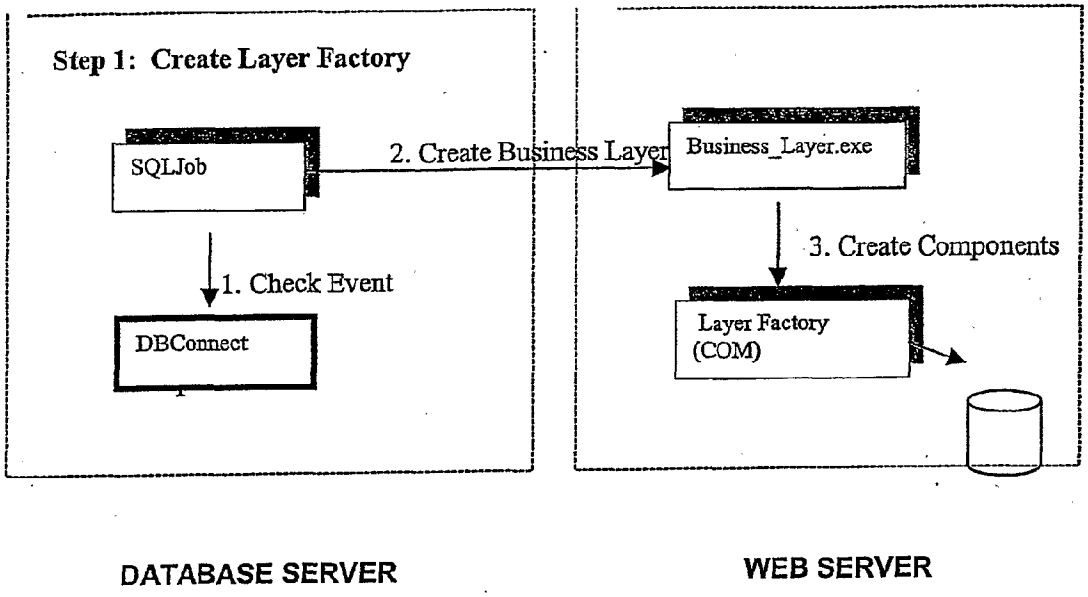


Figure 11

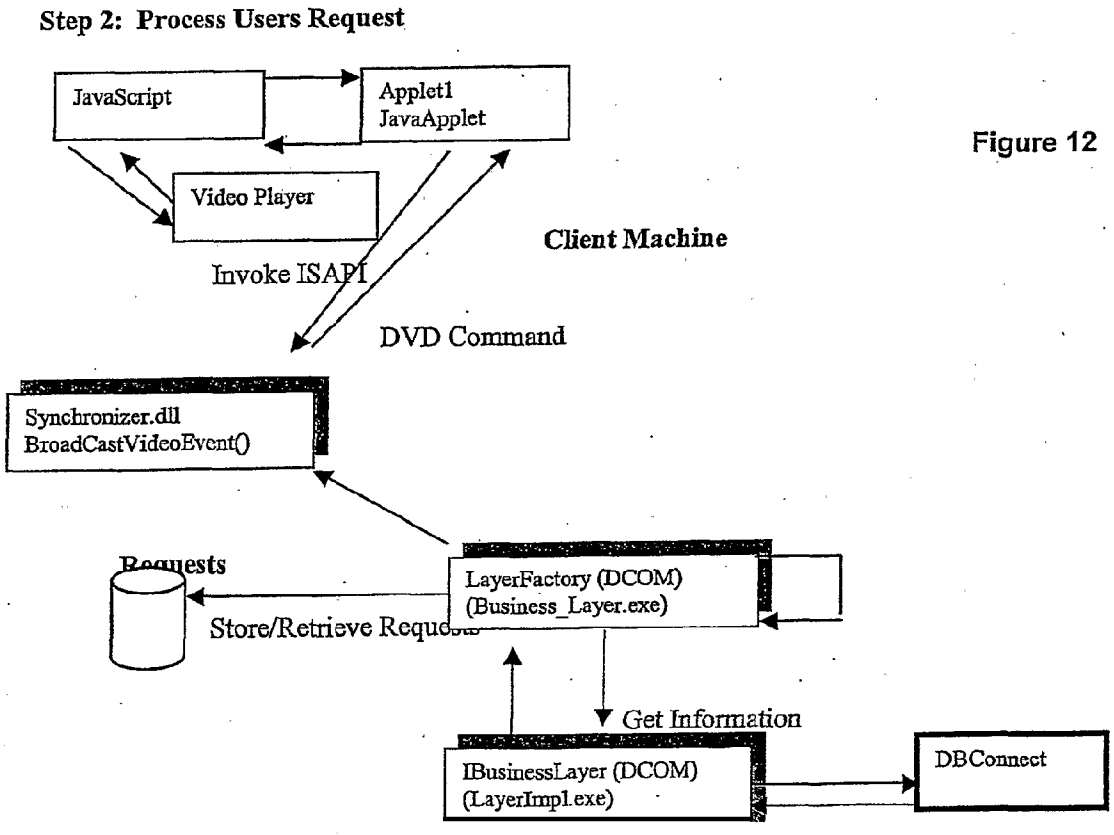


Figure 12

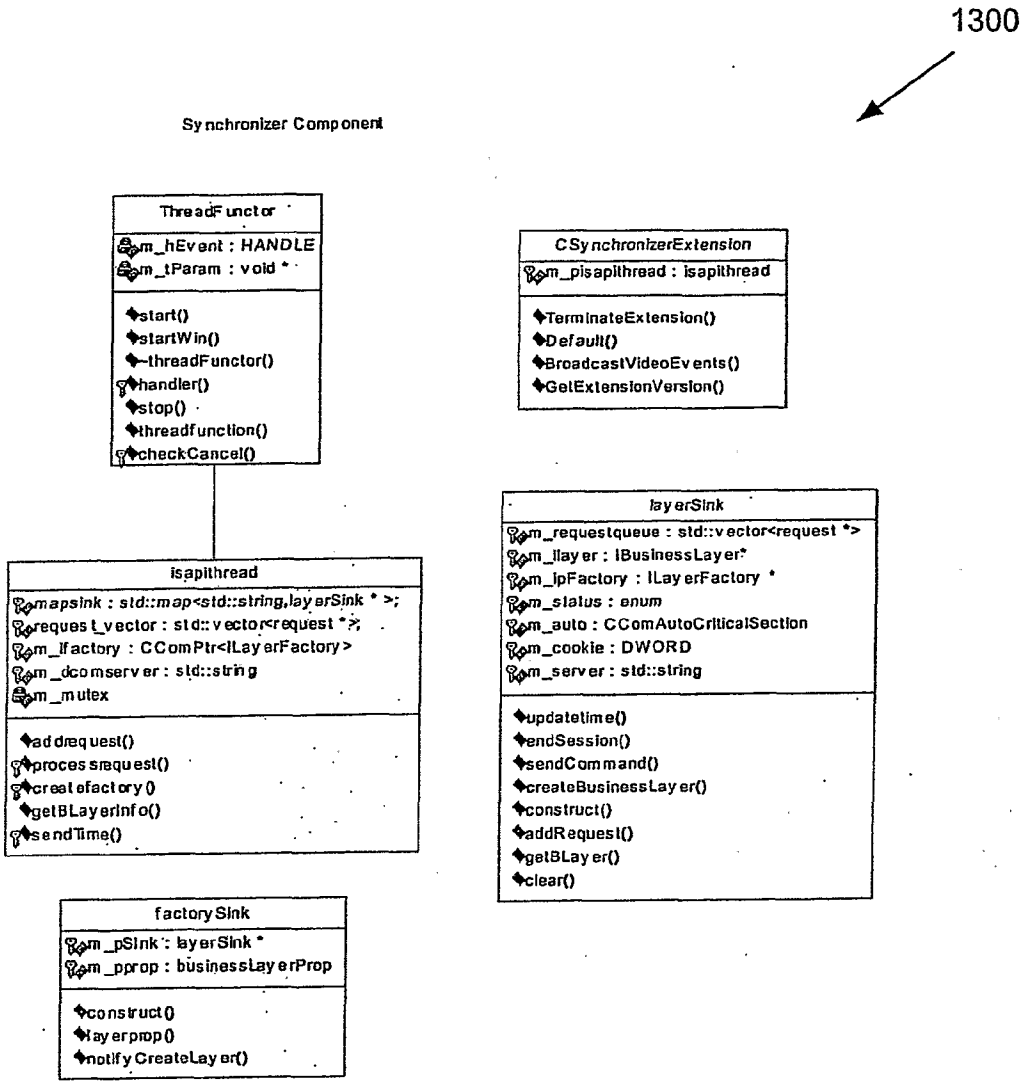

1300
Synchronizer Component
ThreadFunctor
m_hEvent : HANDLE
m_tParam : void *
start()
startWin()
~threadFunctor()
handler()
stop()
threadfunction()
checkCancel()
CSynchronizerExtension
m_pisapithread : isapithread
TerminateExtension()
Default()
BroadcastVideoEvents()
GetExtensionVersion()
isapithread
mapsink : std::map<std::string,layerSink * >;
request_vector : std::vector<request *>;
m_lfactory : CComPtr<ILayerFactory>
m_dcomserver : std::string
m_mutex
addrequest()
processrequest()
createfactory()
getBLayerInfo()
sendTime()
layerSink
m_requestqueue : std::vector<request *>
m_ilayer : IBusinessLayer*
m_ipFactory : ILayerFactory *
m_status : enum
m_auto : CComAutoCriticalSection
m_cookie : DWORD
m_server : std::string
updatetime()
endSession()
sendCommand()
createBusinessLayer()
construct()
addRequest()
getBLayer()
clear()
factorySink
m_pSink : layerSink *
m_pprop : businessLayerProp
construct()
layerprop()
notifyCreateLayer()

Figure 13

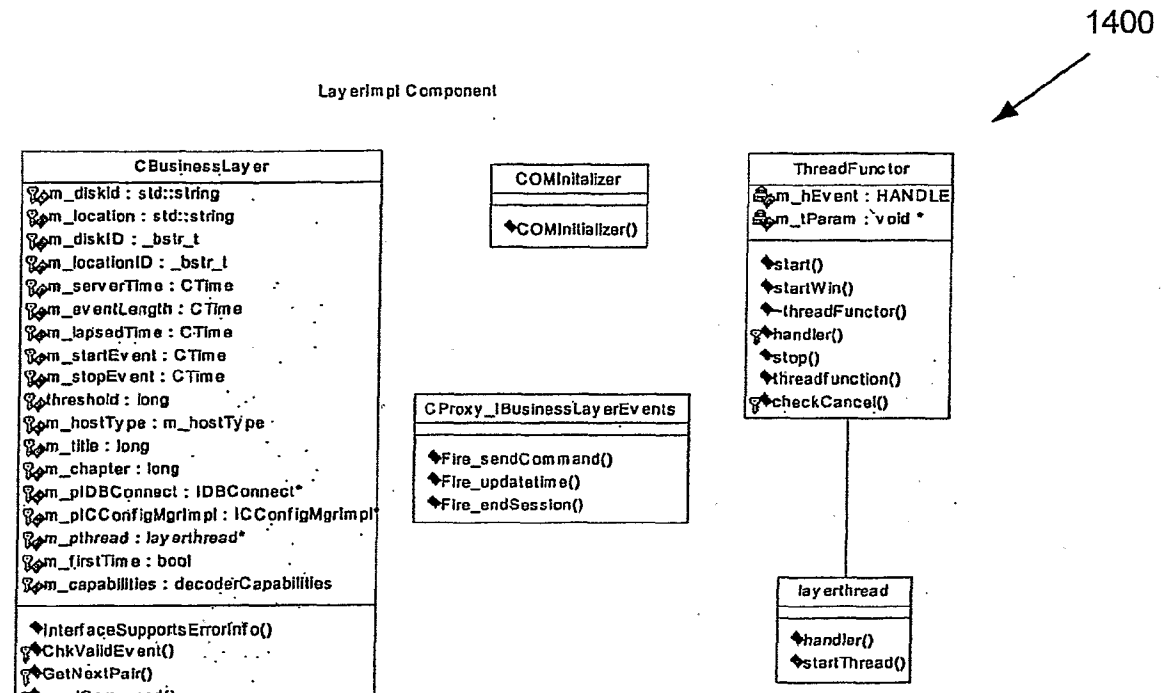

1400
LayerImpl Component
CBusinessLayer
m_diskId : std::string
m_location : std::string
m_diskID : _bstr_t
m_locationID : _bstr_t
m_serverTime : CTime
m_eventLength : CTime
m_lapsedTime : CTime
m_startEvent : CTime
m_stopEvent : CTime
threshold : long
m_hostType : m_hostType
m_title : long
m_chapter : long
m_pIDBConnect : IDBConnect*
m_pICConfigMgrImpl : ICConfigMgrImpl*
m_pthread : layerthread*
m_firstTime : bool
m_capabilities : decoderCapabilities
InterfaceSupportsErrorInfo()
ChkValidEvent()
GetNextPair()
sendCommand()
endSession()
updateTime()
firstdvdCmd()
put_serverTime()
FinalConstruct()
FinalRelease()
get_serverTime()
get_titleProperties()
get_chapterProperties()
get_lapsedTime()
get_eventLength()
TranslateTimePlay()
Initialize()
get_threshold()
put_threshold()
get_stopEvent()
put_stopEvent()
get_startEvent()
put_startEvent()
get_location()
put_location()
get_disk()
put_disk()
COMInitalizer
COMInitializer()
CProxy_IBusinessLayerEvents
Fire_sendCommand()
Fire_updatetime()
Fire_endSession()
ThreadFunctor
m_hEvent : HANDLE
m_tParam : void *
start()
startWin()
~threadFunctor()
handler()
stop()
threadfunction()
checkCancel()
layerthread
handler()
startThread()

Figure 14

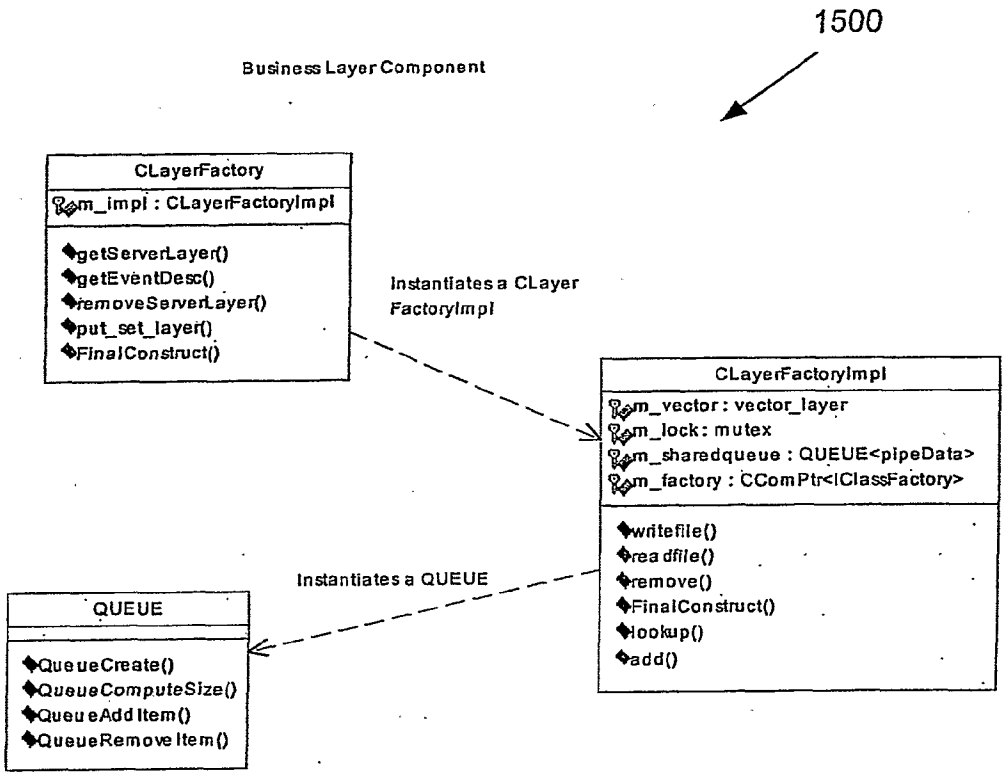

1500
Business Layer Component
CLayerFactory
m_impl : CLayerFactoryImpl
getServerLayer()
getEventDesc()
removeServerLayer()
put_set_layer()
FinalConstruct()
Instantiates a CLayer FactoryImpl
CLayerFactoryImpl
m_vector : vector_layer
m_lock : mutex
m_sharedqueue : QUEUE<pipeData>
m_factory : CComPtr<IClassFactory>
writefile()
readfile()
remove()
FinalConstruct()
lookup()
add()
Instantiates a QUEUE
QUEUE
QueueCreate()
QueueComputeSize()
QueueAddItem()
QueueRemoveItem()

Figure 15

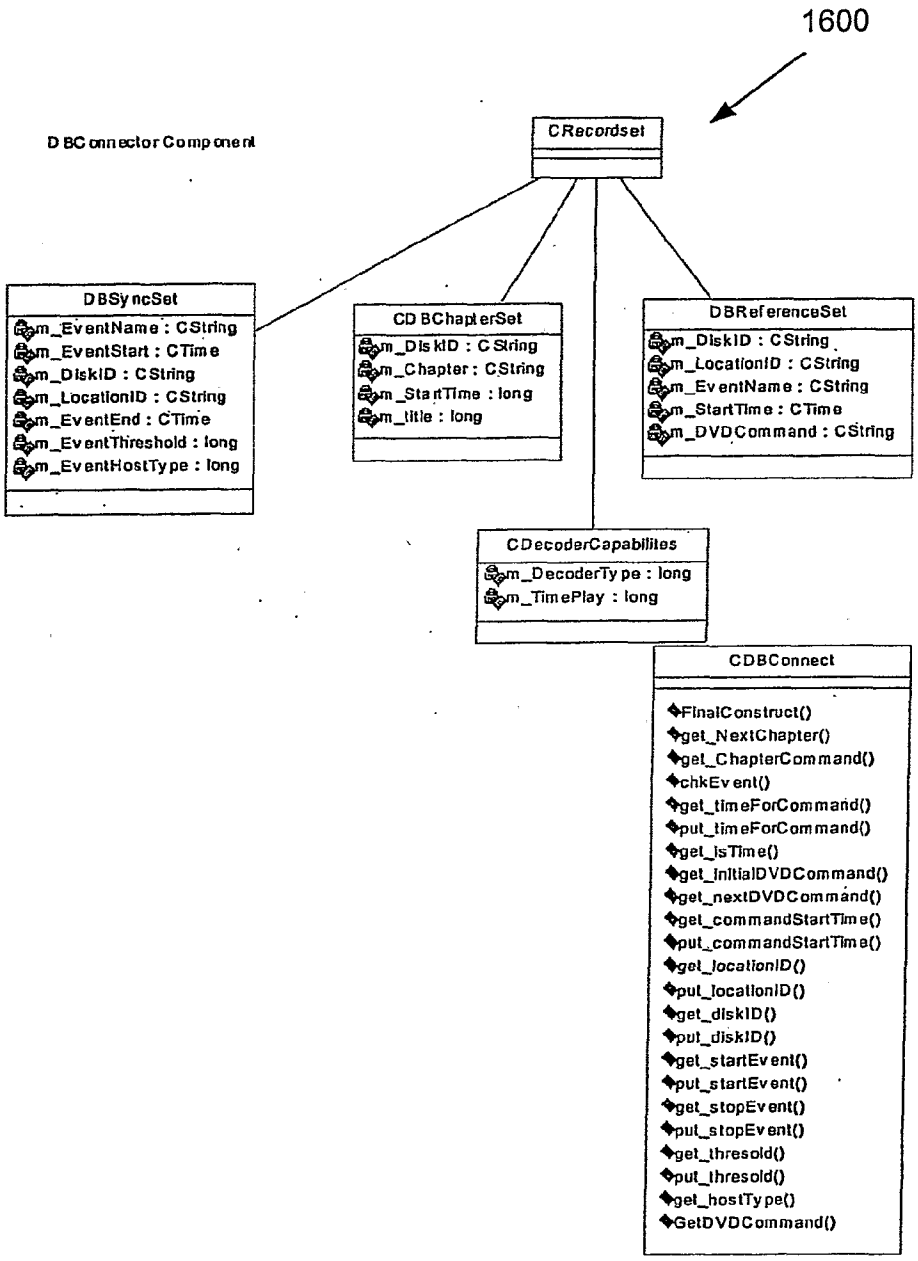

1600
DBConnector Component
CRecordset
DBSyncSet
m_EventName : CString
m_EventStart : CTime
m_DiskID : CString
m_LocationID : CString
m_EventEnd : CTime
m_EventThreshold : long
m_EventHostType : long
CDBChapterSet
m_DiskID : CString
m_Chapter : CString
m_StartTime : long
m_title : long
DBReferenceSet
m_DiskID : CString
m_LocationID : CString
m_EventName : CString
m_StartTime : CTime
m_DVDCommand : CString
CDecoderCapabilites
m_DecoderType : long
m_TimePlay : long
CDBConnect
FinalConstruct()
get_NextChapter()
get_ChapterCommand()
chkEvent()
get_timeForCommand()
put_timeForCommand()
get_isTime()
get_initialDVDCommand()
get_nextDVDCommand()
get_commandStartTime()
put_commandStartTime()
get_locationID()
put_locationID()
get_diskID()
put_diskID()
get_startEvent()
put_startEvent()
get_stopEvent()
put_stopEvent()
get_thresold()
put_thresold()
get_hostType()
GetDVDCommand()

Figure 16

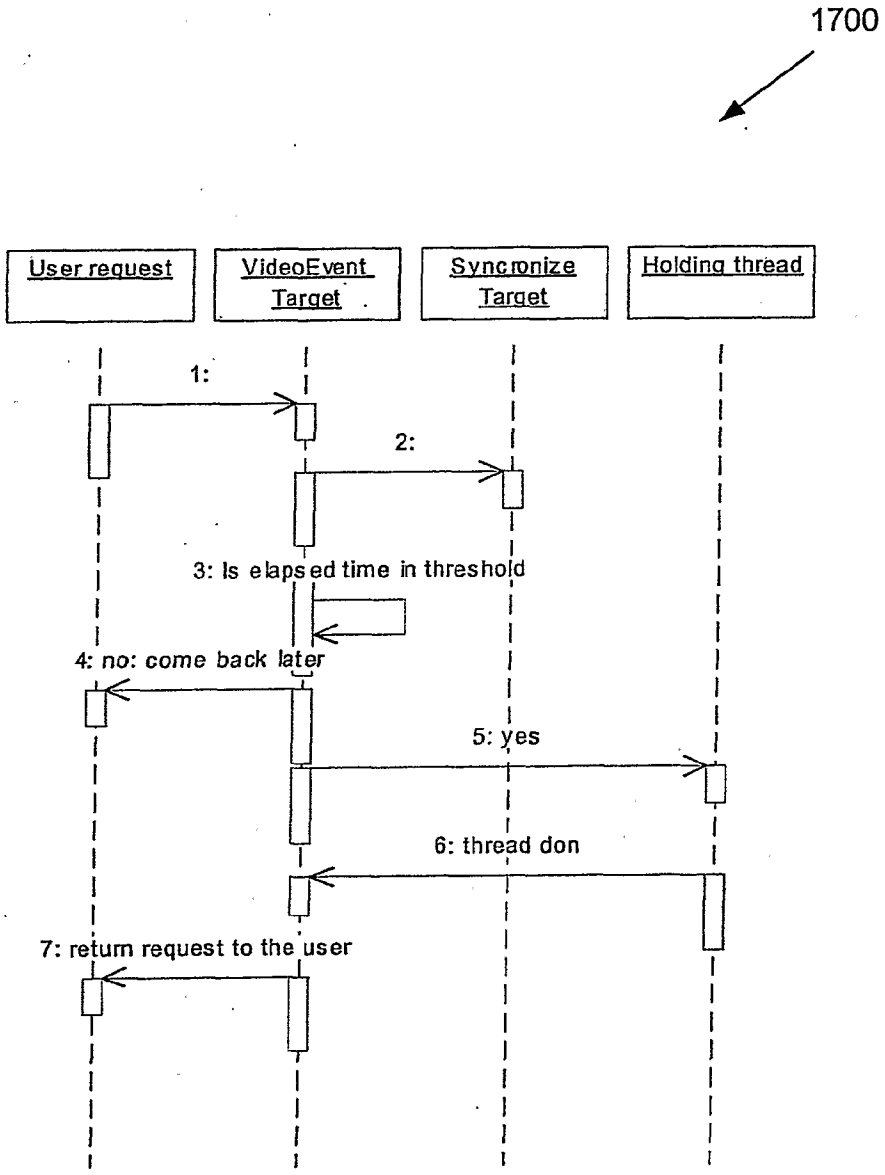

1700
User request
VideoEvent Target
Syncronize Target
Holding thread
1:
2:
3: Is elapsed time in threshold
4: no: come back later
5: yes
6: thread don
7: return request to the user

Figure 17

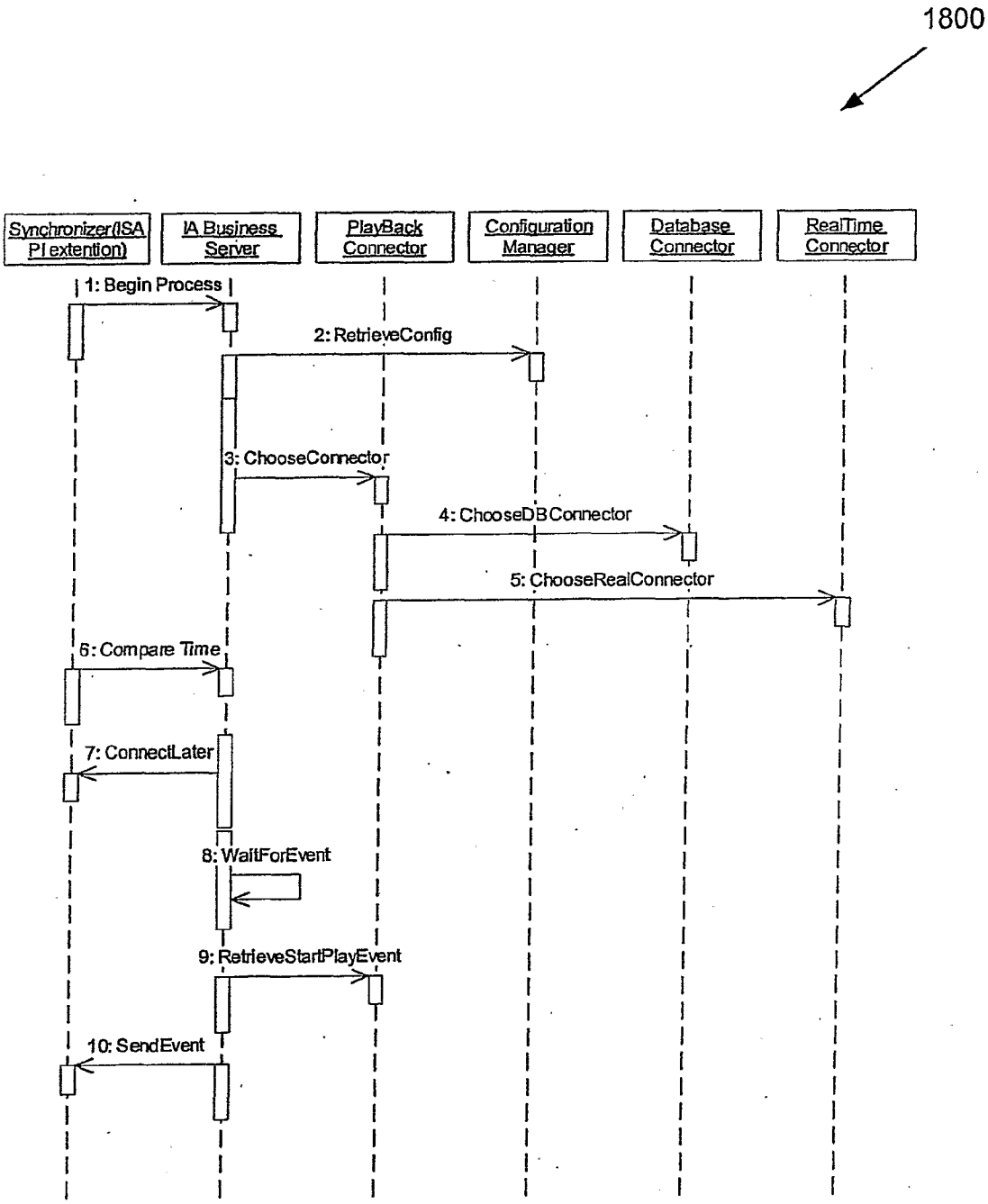

1800
Synchronizer(ISA PI extention)
IA Business Server
PlayBack Connector
Configuration Manager
Database Connector
RealTime Connector
1: Begin Process
2: RetrieveConfig
3: ChooseConnector
4: ChooseDBConnector
5: ChooseRealConnector
6: Compare Time
7: ConnectLater
8: WaitForEvent
9: RetrieveStartPlayEvent
10: SendEvent

Figure 18

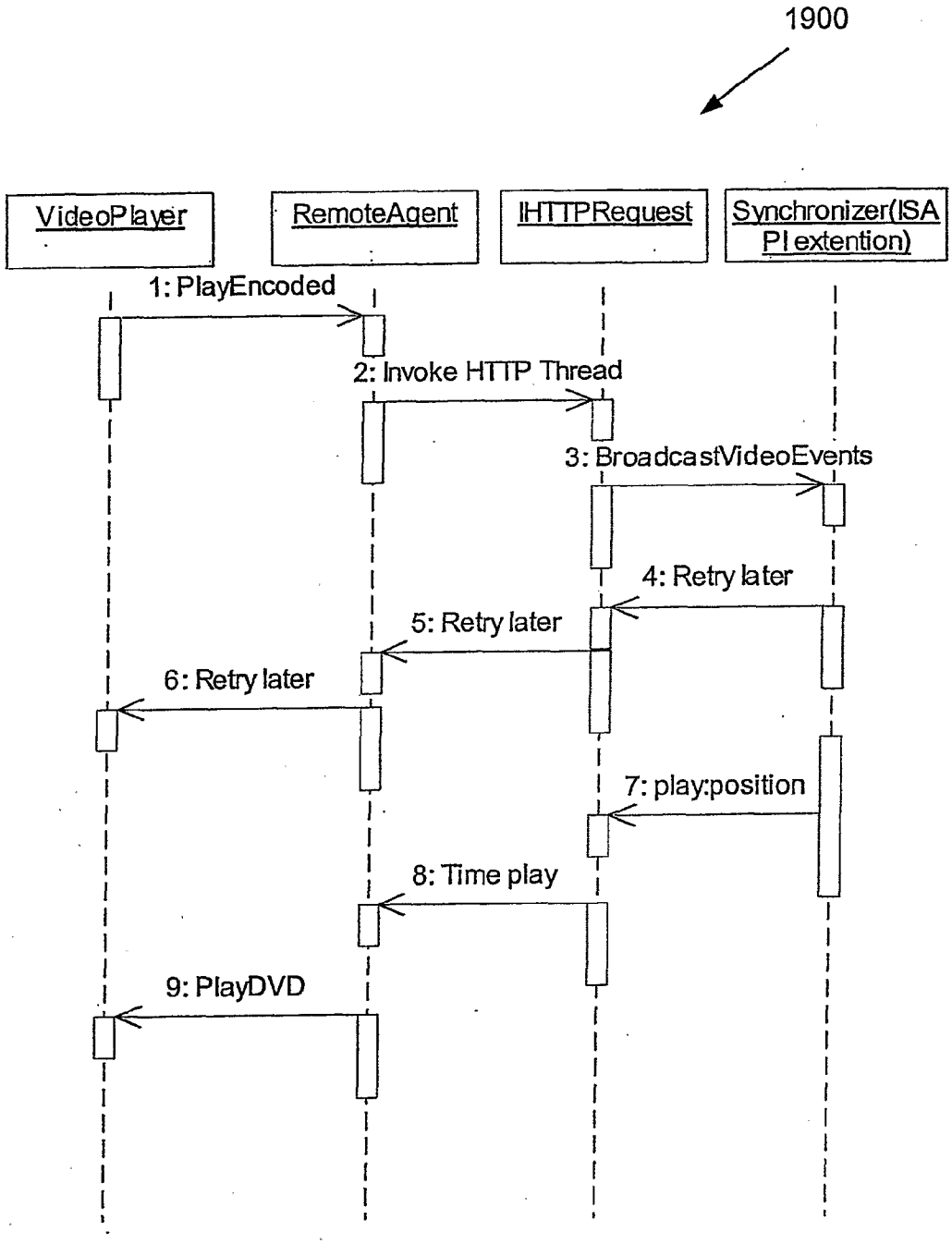

1900
VideoPlayer
RemoteAgent
IHTTPRequest
Synchronizer(ISA PI extention)
1: PlayEncoded
2: Invoke HTTP Thread
3: BroadcastVideoEvents
4: Retry later
5: Retry later
6: Retry later
7: play:position
8: Time play
9: PlayDVD

Figure 19

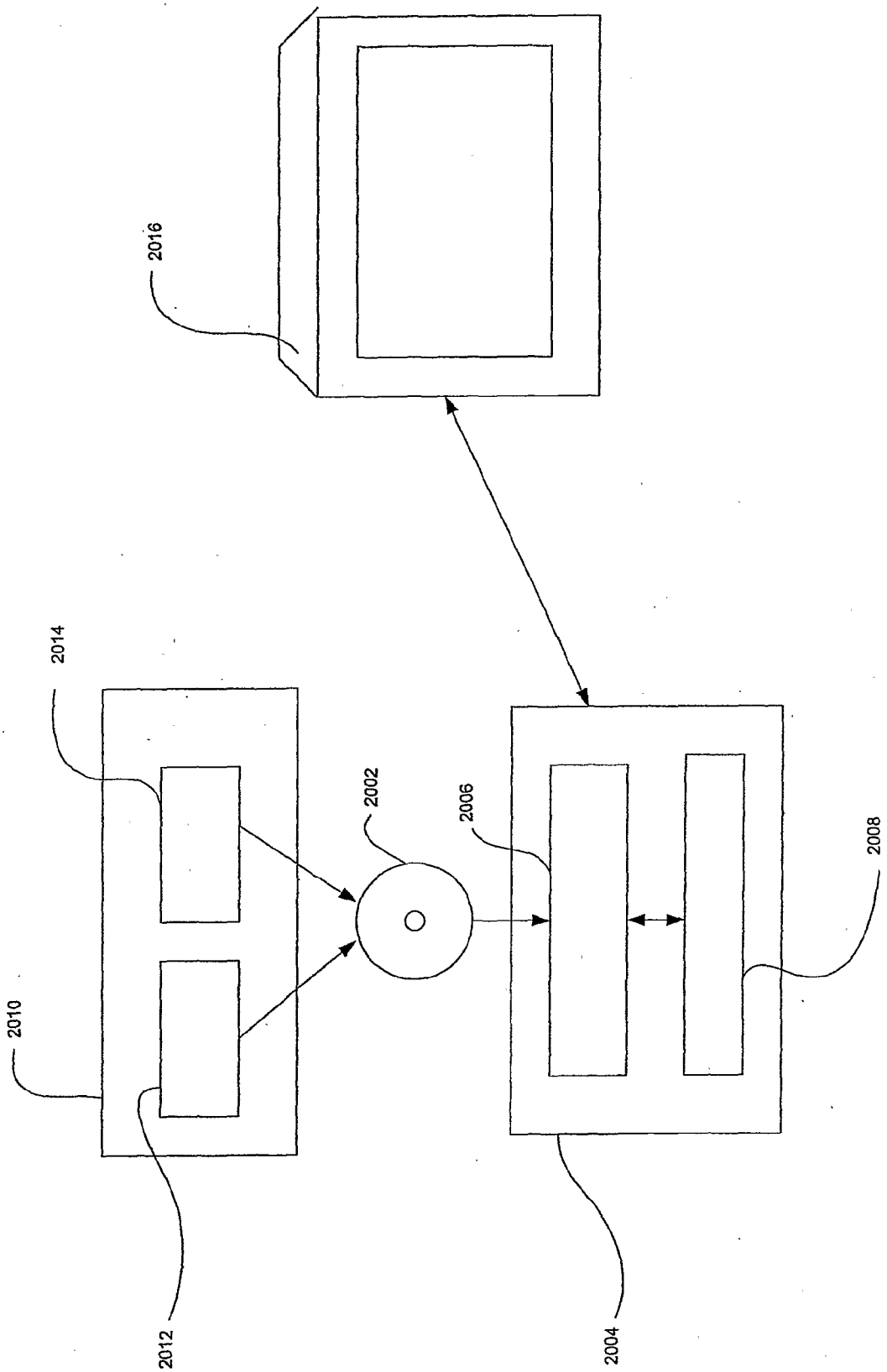


Figure 20

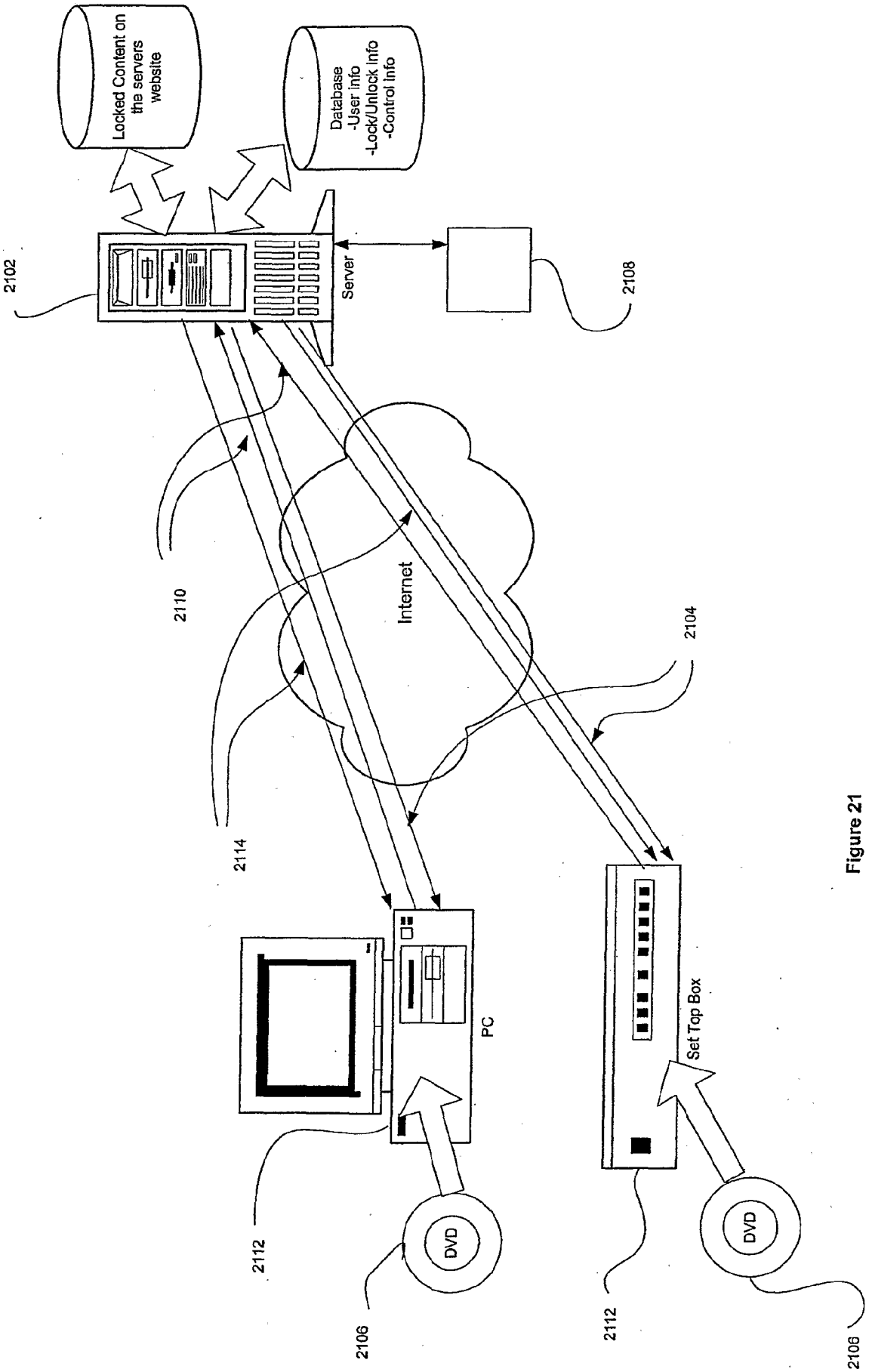
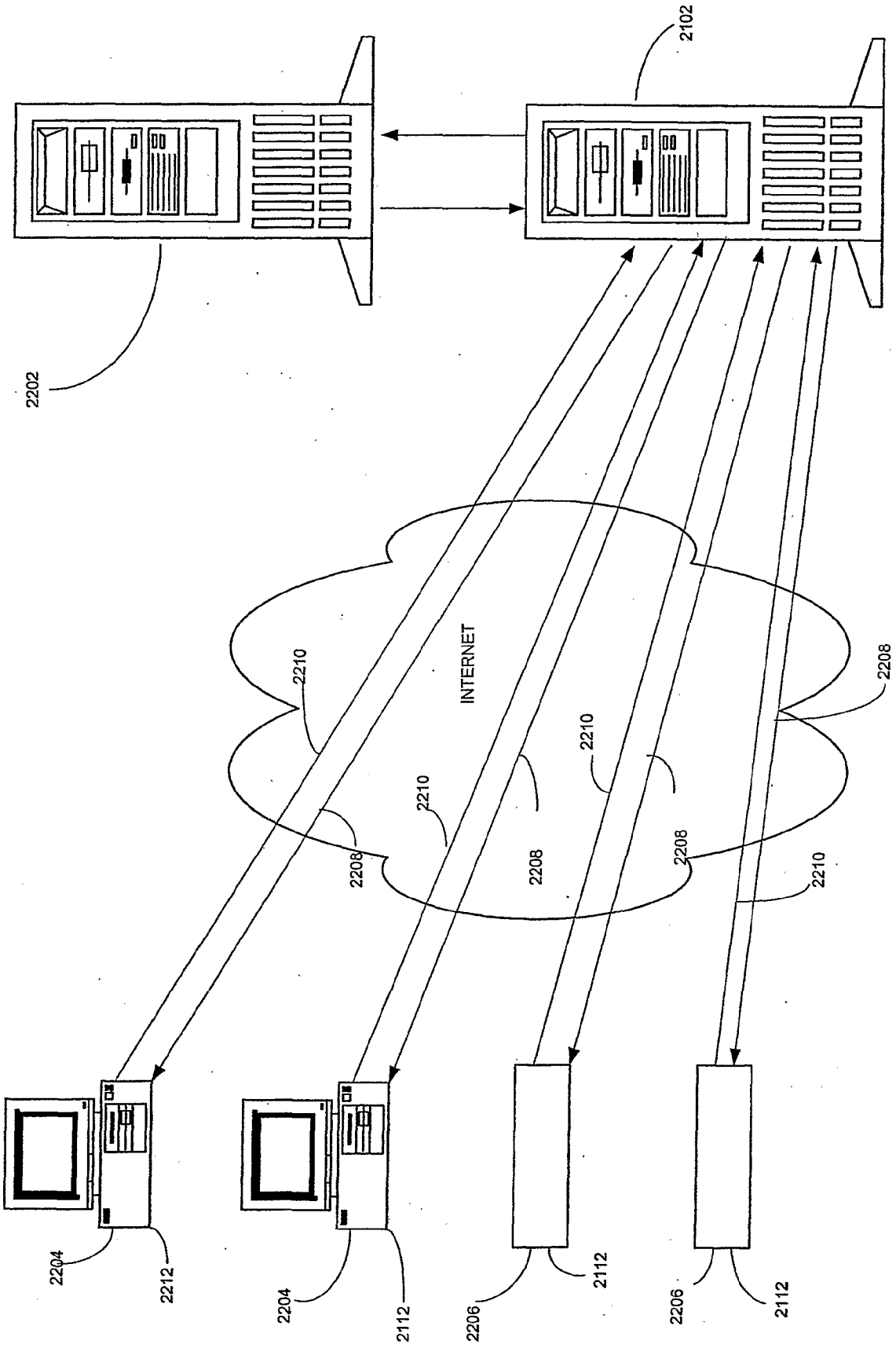

Locked Content on the servers website
Database
-User info
-Lock/Unlock info
-Control info
2102
Server
2108
2110
Internet
2104
2114
PC
Set Top Box
DVD
DVD
2112
2112
2106
2106

Figure 21


2102
2202
INTERNET
2210
2208
2204
2212
2112
2206

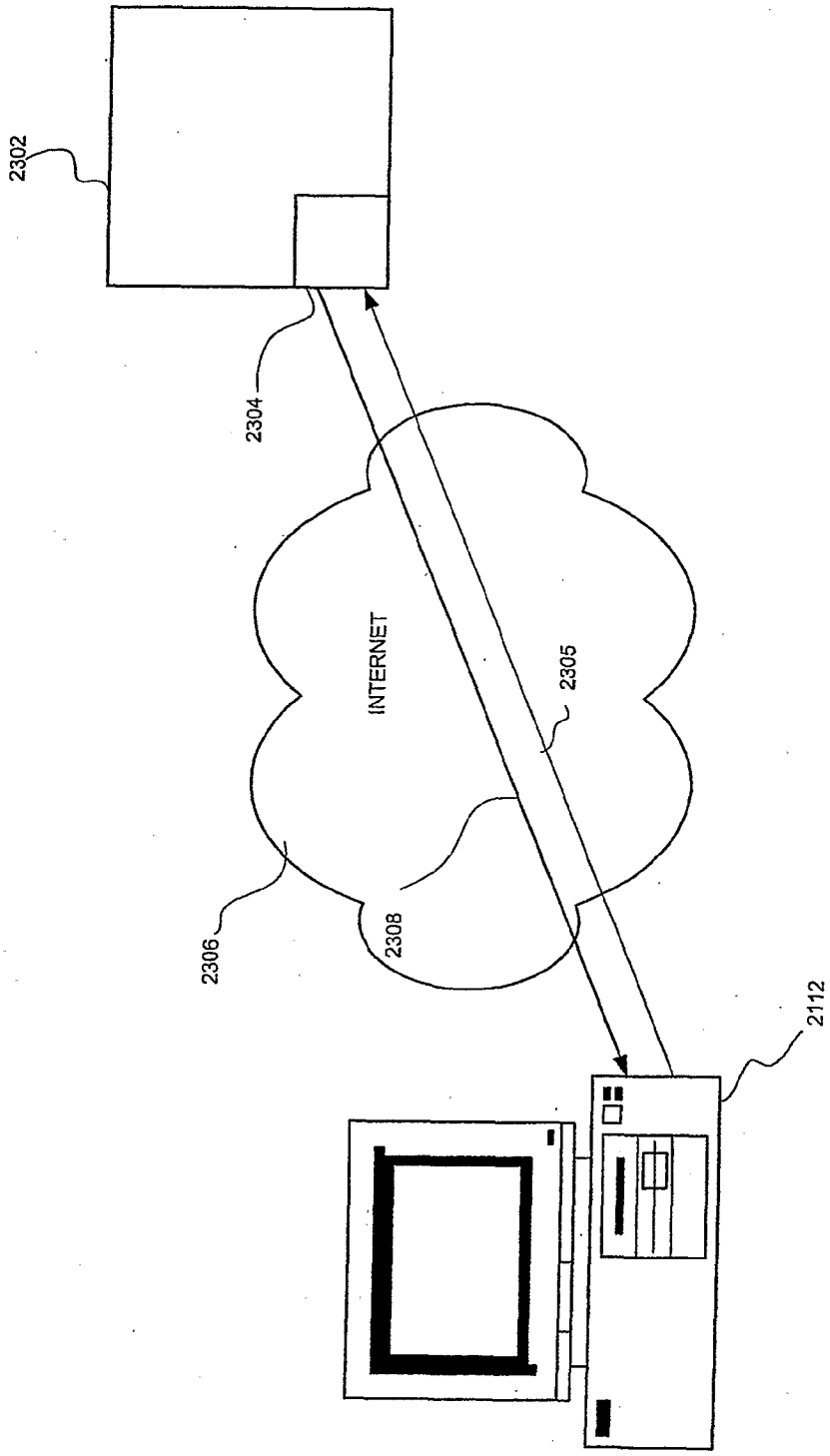


Figure 23

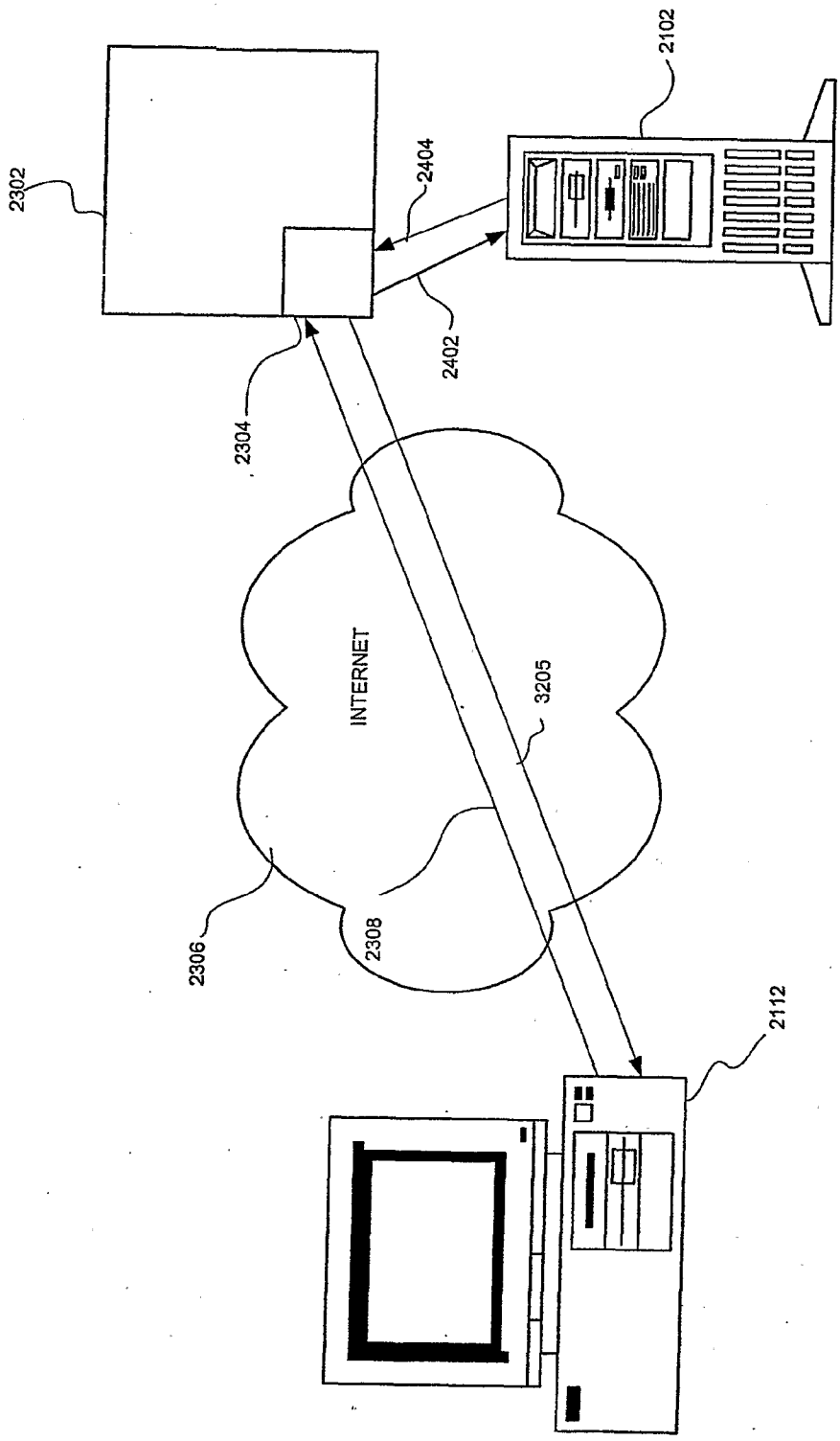


Figure 24

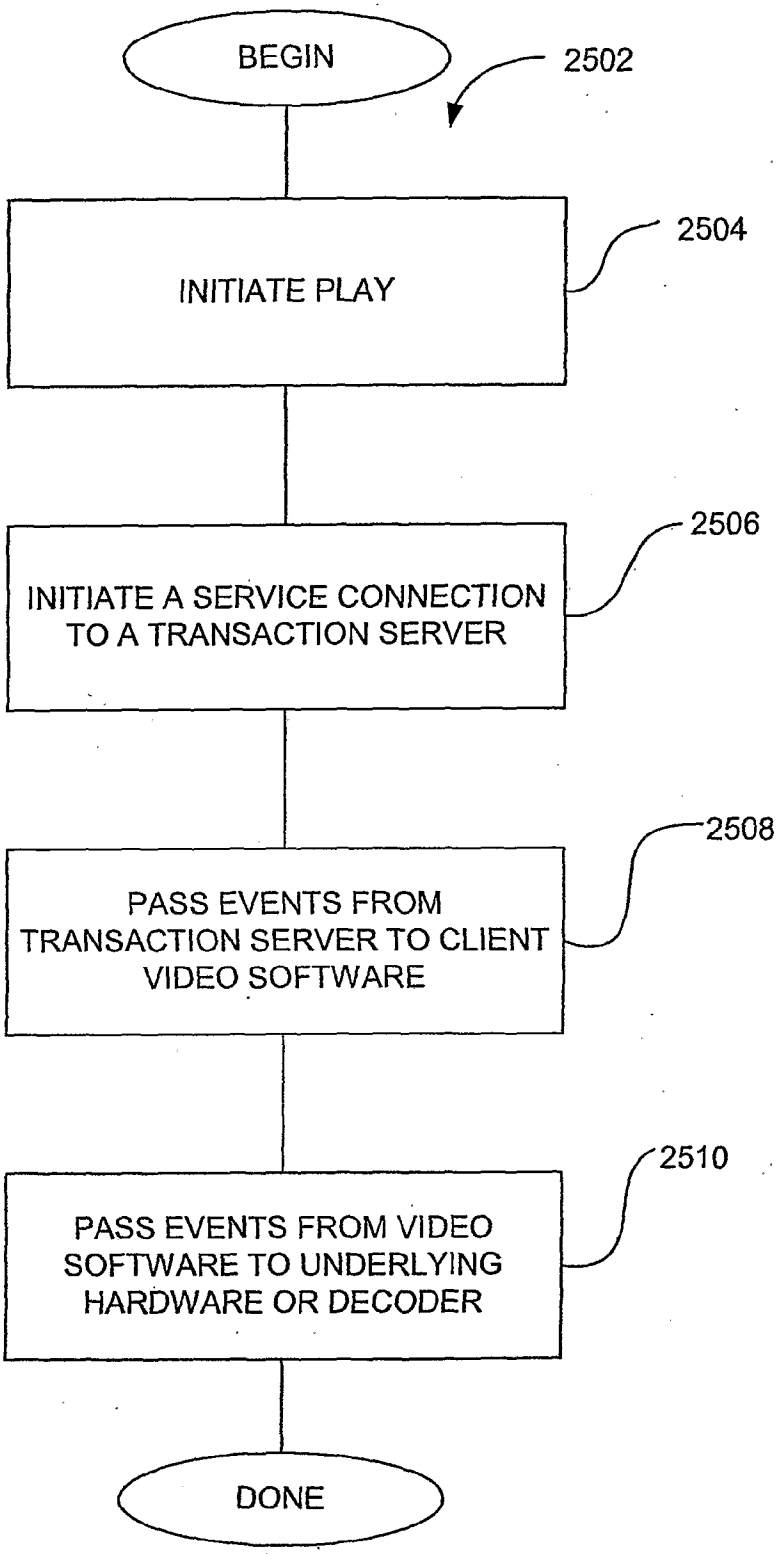

BEGIN
2502
INITIATE PLAY
2504
INITIATE A SERVICE CONNECTION TO A TRANSACTION SERVER
2506
PASS EVENTS FROM TRANSACTION SERVER TO CLIENT VIDEO SOFTWARE
2508
PASS EVENTS FROM VIDEO SOFTWARE TO UNDERLYING HARDWARE OR DECODER
2510
DONE

Figure 25

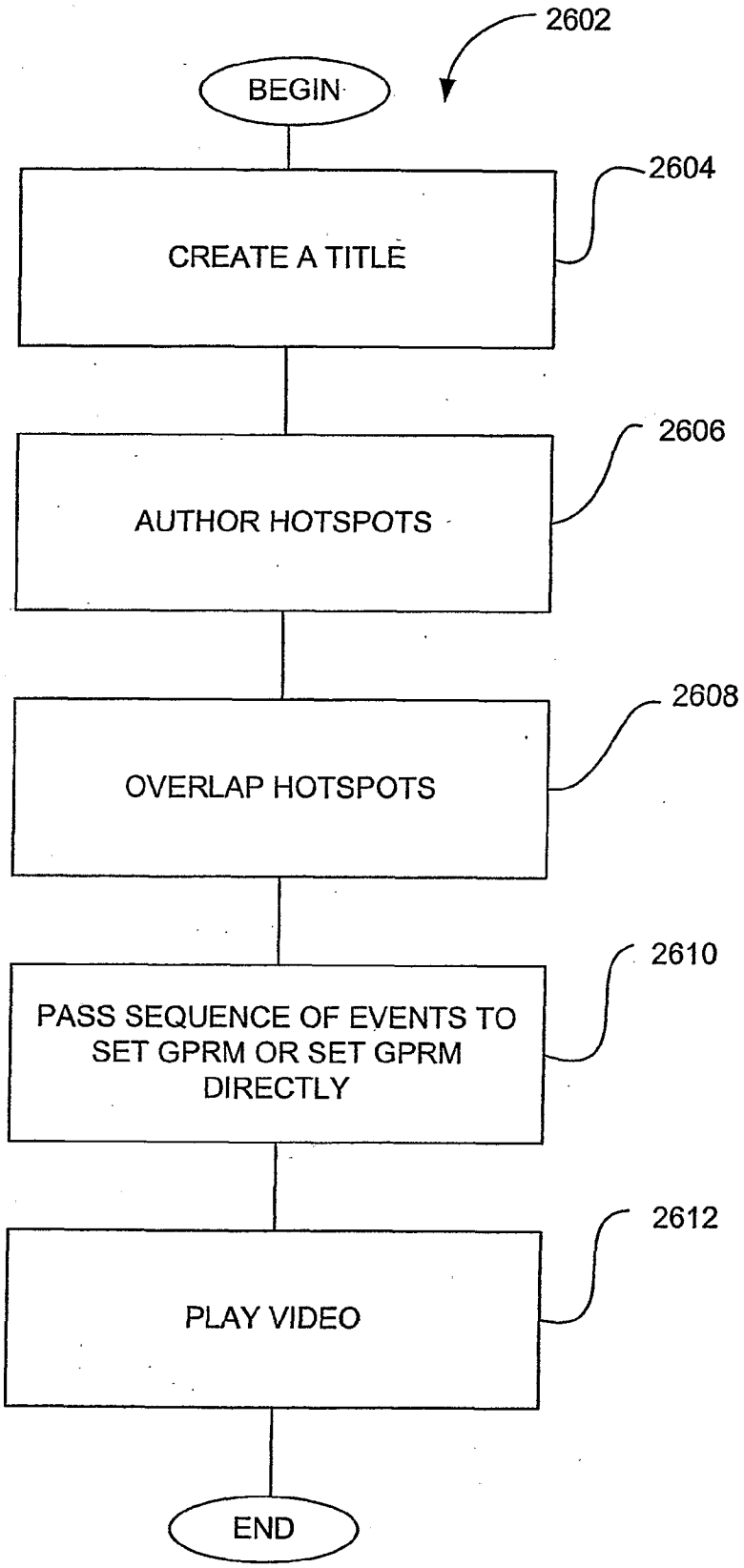

2602
BEGIN
2604
CREATE A TITLE
2606
AUTHOR HOTSPOTS
2608
OVERLAP HOTSPOTS
2610
PASS SEQUENCE OF EVENTS TO SET GPRM OR SET GPRM DIRECTLY
2612
PLAY VIDEO
END

Figure 26

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/02143

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :H04L 9/00; G06F 13/00; H04L 9/00, 9/08, 9/32
US CL :380/203, 277; 709/217, 218
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 380/203, 277; 709/217, 218

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 6,035,329 A (MAGES ET AL.) 07 MARCH 2000, COL.2-5.	1-185
Y, P	US 6,185,306 B1 (MAGES ET AL.) 06 FEBRUARY 2001, COL.3-10.	1-185
Y	US 5,937,164 A (MAGES ET AL.) 10 AUGUST 1999, COL.3-8.	1-185
Y	US 5,892,825 A (MAGES ET AL.) 06 APRIL 1999, COL.3-8.	1-185

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
02 APRIL 2001	26 APR 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer GLEN BURGESS James R. Matthews Telephone No. (703) 305-4792

Form PCT/ISA/210 (second sheet) (July 1998)★

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 July 2001 (26.07.2001)

PCT

(10) International Publication Number
WO 01/054344 A1

(51) International Patent Classification[7]: H04L 9/00, G06F 13/00, H04L 9/00, 9/08, 9/32

(21) International Application Number: PCT/US01/02143

(22) International Filing Date: 22 January 2001 (22.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/488,345	20 January 2000 (20.01.2000)	US
09/488,337	20 January 2000 (20.01.2000)	US
09/488,143	20 January 2000 (20.01.2000)	US
09/488,613	20 January 2000 (20.01.2000)	US
09/488,155	20 January 2000 (20.01.2000)	US
09/489,600	20 January 2000 (20.01.2000)	US
09/488,614	20 January 2000 (20.01.2000)	US
09/489,601	20 January 2000 (20.01.2000)	US
09/489,596	20 January 2000 (20.01.2000)	US
09/499,247	7 February 2000 (07.02.2000)	US

(71) Applicant: INTERACTUAL TECHNOLOGIES, INC. [US/US]; 100 Century Center Court, #205, San Jose, CA 95112 (US).

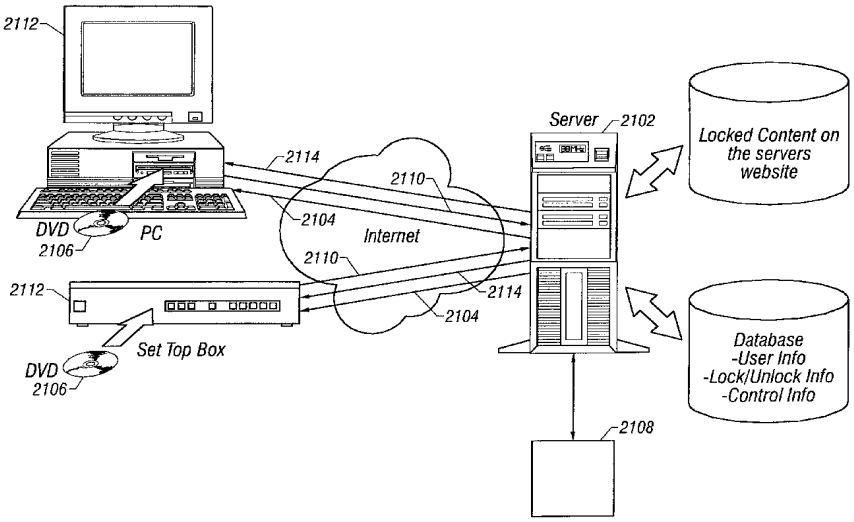
(72) Inventors: COLLART, Todd, R.; 206 Arbuelo Way, Los Altos, CA 94022 (US). LAMKIN, Allan, B.; 4282 Farley Lane, San Diego, CA 92122 (US). GETSIN, Evgeniy, M.; 11556 E. Powers Avenue, Englewood, CO 80111 (US). LEWIS, Michael, J.; 2058 South Kenton Court, Aurora, CO 80014 (US).

(74) Agents: RANNEY, Kathleen, A. et al.; Fitch, Even, Tabin & Flannery, Suite 1600, 120 South LaSalle Street, Chicago, IL 60603 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR REMOTE CONTROL AND NAVIGATION OF LOCAL CONTENT



(57) Abstract: A system, method and article of manufacture are provided for remotely controlling content stored on a local computer connected to a network system such as the Internet. The embodiment allows content such as DVD (2106) Video content to be locked so that play can only be accomplished through verification of user identity and also allows augmentation and supplementation of the content provided by the video from a remote server (2102). In addition, upon verification of user client credentials, the present embodiment allows the locally stored content to be supplemented with additional content delivered over the network system. Furthermore, the present embodiment allows content stored locally on a client device (2112) to be remotely controlled to play synchronously on a plurality of client devices.

WO 01/054344 A1

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(48) Date of publication of this corrected version:
7 November 2002

(15) Information about Correction:
see PCT Gazette No. 45/2002 of 7 November 2002, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR REMOTE CONTROL AND NAVIGATION OF LOCAL CONTENT

BACKGROUND

Multimedia computer systems have become increasingly popular over the last several years due to their versatility and their interactive presentation style. A multimedia computer system can be defined as a computer system having a combination of video and audio outputs for presentation of audio-visual displays. A modern multimedia computer system typically includes one or more storage devices such as an optical drive, a CD-ROM, a hard drive, a videodisc, or an audiodisc, and audio and video data are typically stored on one or more of these mass storage devices. Audio and video data for a multimedia display may also be stored in separate computer systems that are networked together. In this instance, the computer system presenting the multimedia display would receive a portion of the necessary data from the other computer system via the network cabling.

Such audio and video data or content is often stored on media such as CD-ROM or digital video disc (DVD). However, once a vendor has delivered such content to a customer, the vendor loses any practical control over the product. Even if the product is delivered under license rather than out right sale, it has traditionally been difficult to prevent a customer from copying the content or providing the content to any number of friends so that they might illegally copy the content.

Another problem which arises from the vendors loss of control of the content maintenance and updating of the software. If content is to be added or modified, the vendor must send a new disc to the customer. In addition, the vendor can not control the amount of data which the customer can access. In other words, once the disc is delivered, the customer will have access to all of the content on the disc and only that content. Time sensitive content, such as advertising, can become obsolete but will still be accessible on the disc.

Therefore, there remains a need for a system method or apparatus allowing flexible control of content delivered to a client. Such a system, method or apparatus would preferably allow content to be initially delivered on a traditional recording medium such as a CD-ROM or DVD but would allow a vendor to remote control the

access of a user to the content stored thereon. Furthermore, such a system would preferably allow a vendor supplement and/or modify the content and could allow the vendor to limit a client's access to certain portions of the locally stored content if desired. Furthermore, remote control of navigation would be preferably and could facilitate simultaneous access by a controlled number of multiple clients if desired.

SUMMARY OF THE INVENTION

A system, method and article of manufacture are provided for synchronizing an event on a plurality of a client apparatuses. First, an event is stored in a memory storage device. The client apparatuses and a host computer are adapted to be connected to a network. In operation, information is transmitted from the host computer to the memory storage device utilizing the network. This allows for the simultaneous playback of the event on each of the client apparatuses.

In a another further embodiment, the invention can be characterized as a system, method and article of manufacture for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses. Initially, an event is stored in memory on at least one of a plurality of client apparatuses. These client apparatuses and a host computer are adapted to be connected to a network during use. Information is stored on the host computer for allowing the simultaneous playback of the event on each of the client apparatuses. In operation, the information may be downloaded utilizing the network for playback after the simultaneous playback of the event from the memory.

In a supplemental embodiment, the present invention can be characterized as, a system, method and article of manufacture for providing overlays during a synchronized event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network. Further, an event may be simultaneously played back on the client apparatuses utilizing the network. During the playback of the event, visual and/or audio material may also be overlaid on the event based on input received utilizing the network.

In an additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for delayed synchronization of an event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network and an event is stored in memory on the client apparatuses.

The event is then simultaneously played back on the client apparatuses utilizing the network. During the simultaneous playback, a request may be received from one of the client apparatuses for that particular client to be included in the synchronized event. In response to the request, information is transmitted to the requesting client apparatus utilizing the network. This information is adapted for identifying a location in the memory where the event is currently being played back on the remaining client apparatuses. This allows the simultaneous playback of the event on the requesting client apparatus. In one embodiment of the present invention, the request may be received utilizing the network.

In another embodiment, the present invention can be characterized as, a system, method and article of manufacture for synchronizing an event on a plurality of client apparatuses. First, a plurality of client apparatuses are connected via a network. Next, an application program is embedded on a site on the network. In use, information is requested from a server on the network utilizing the application program. Such information relates to an event to be played back simultaneously on the client apparatuses. In response to such request, a script is received for displaying the information.

In a further additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses. First, a request is received utilizing a network for viewing an event. Next, the request is queued in memory. In response to the request, an object is created which is adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal. The object is sent to one of the client apparatuses utilizing the network for being stored therein.

In yet another embodiment, the present invention can be characterized as, system, method and article of manufacture for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses. First, various values are determined including a current time, a start time when an event is to start, and a stop time when the event is to end. Thereafter, a length of the event is calculated based on the start time and the stop time. If any portion of the length of the event takes place during a predetermined

threshold period, a command is stored in memory. Further, a loop is created at the start time during which a lapsed time of the event is tracked.

In still yet an additional embodiment, the present invention can be characterized as a system, method and article of manufacture for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses. First, a plurality of events are stored in memory on a plurality of client apparatuses. The events each have a unique identifier associated therewith and which are stored in the memory. In operation, the client apparatuses are adapted to be coupled to a host computer via a network. The identifier of the event which is stored in the memory of the client apparatuses is then retrieved utilizing the network. Such identifier is subsequently compared with an identifier of a scheduled event. If the comparison renders a match, the playback of the event is begun on each of the client apparatuses.

In a further additional embodiment, the present invention can be characterized as, a system, method and article of manufacture are for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event. A type of the playback device of each of the client apparatuses is first identified. A command associated with the identified type of the playback device is then looked up in a table. Thereafter, the command is sent to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

In another additional embodiment, the present invention can be characterized as, a system, method and article of manufacture for remotely controlling local content for local access and use by a client device. An input is received from the client which can allow a transaction server to identify the client. Once the client has been identified a command can be sent to the client which controls the manner in which the client device can use and access the local content.

The local content can be embodied on a digital video disk, and commands generated by the transaction server can be in the form of an unlock sequence which allows the client device to access and use the content stored on the disk. In addition, commands from the transaction server can be used to navigate the content stored on the disk and can even supplement the content stored thereon. The transaction server

can, in response to a client identification, unlock content stored remotely on the transaction server, allowing content to be easily maintained and updated remotely at a single transaction site without having to replace many DVD disks being used by many different clients.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood when consideration is given to the following detailed description thereof. Such description makes reference to the annexed drawings wherein:

Figure **1** is a schematic diagram of a hardware implementation of one embodiment of the present invention;

Figure **2** illustrates a flowchart delineating a method for synchronizing an event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **3** illustrates a flowchart delineating a method for storing synchronization information for subsequent playback of an event in accordance with one embodiment of the present invention;

Figure **4** illustrates a flowchart setting forth a method for providing overlays during a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **5** illustrates a flow diagram for delayed synchronization of an event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **6** illustrates a flow diagram for providing information on a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **7** illustrates a method for creating a synchronizer object in order to playback an event simultaneously on a plurality of client apparatuses in accordance with one embodiment of the present invention;

Figure **8** illustrates a flowchart for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses in accordance with one embodiment of the present invention;

Figure **9** is a flowchart delineating a method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses in accordance with one embodiment of the present invention;

Figure **10** shows a flowchart delineating a technique for interfacing a plurality of different types of playback devices of client apparatuses which are networked to simultaneously playback an event in accordance with one embodiment of the present invention;

Figure **11** illustrates the manner in which a layer factory is created in accordance with one embodiment of the present invention;

Figure **12** illustrates the manner in which user requests are processed in accordance with one embodiment of the present invention;

Figures **13-16** illustrate various class/component diagrams in accordance with one embodiment of the present invention;

Figure **17** illustrates a logical sequence diagram in accordance with one embodiment of the present invention;

Figure **18** illustrates a logical sequence diagram that shows server side collaboration in accordance with one embodiment of the present invention;

Figure **19** illustrates a logical sequence diagram showing client side collaboration in accordance with one embodiment of the present invention;

Figure **20** is a schematic diagram of a process according to an embodiment of the present invention;

Figure **21** is a schematic diagram illustrating an embodiment of the present invention;

Figure **22** is a schematic diagram illustrating another embodiment of the present invention;

Figure **23** is a schematic diagram illustrating yet another embodiment of the present invention;

Figure **24** is a schematic diagram illustrating still another embodiment of the present inveniton;

Figure **25** is a flowchart illustrating a method for carrying out the present invention; and

Figure **26** is a flowchart illustrating a method for carrying out an aspect of the present inveniton.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figures **1-26** illustrate a system for remotely controlling content stored locally on a client apparatus. Prior to use, an event is stored in memory on at least one client apparatus. Such client apparatus is adapted to be connected to a network along with a host computer(s). In operation, information is transmitted from the host computer to the at least one client apparatus utilizing the network. This information controls playback of the event stored on locally on the client computer.

In various embodiments, the client apparatus may take the form of a computer, television, stereo, home appliance, or any other types of devices. In one embodiment, the client apparatuses and the host computer each include a computer such as an IBM compatible computer, Apple Macintosh computer or UNIX based workstation.

A representative hardware environment is depicted in Figure **1**, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **110**, such as a microprocessor, and a number of other units interconnected via a system bus **112**. The workstation shown in Figure **1** includes a Random Access Memory (RAM) **114**, Read Only Memory (ROM) **116**, an I/O adapter **118** for connecting peripheral devices such as disk storage units **120** (i.e. DVD playback device) to the bus **112**, a user interface adapter **122** for connecting a keyboard **124**, a mouse **126**, a speaker **128**, a microphone **132**, and/or other user interface devices such as a touch screen (not shown) to the bus **112**, communication adapter **134** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **136** for connecting the bus **112** to a display device **138**. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present embodiment may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using Java, C, and the C++ language and utilizes object oriented programming methodology. A preferred embodiment of the

embodiment utilizes HyperText Markup Language (HTML), including Java, JavaScript (ECMAScript), ActiveX, or the like, to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the server. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H, Frystyk, T. Berners-Lee, J. Gettys and J.C. Mogul, "Hypertext Transfer Protocol -- HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML is an application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

Synchronization Overview

Figure **2** illustrates a flowchart delineating a method for synchronizing an event on a plurality of client apparatuses. First, in operation **200**, an event is stored in memory on at least one of the client apparatuses. In various embodiments, the memory may take the form of an electromagnetic medium, or any type of optical storage device, i.e. CD-audio. In a primary aspect of the present embodiment, the memory includes a digital video disc (DVD) (audio or video). Further, for reasons that will soon become apparent, the information includes chapter information associated with the DVD. In such embodiment where the memory is portable, the user may be required to purchase the memory, i.e. DVD, in order to participate in a synchronized event, thus increasing the sale of DVD's.

It should be noted that the event need not be necessarily stored in memory on all of the client apparatuses, but rather stored on one or some of the client apparatuses and streamed to the remaining client apparatuses at variant rates. This may be feasibly accomplished if the client apparatus(es) containing the stored event has a high-bandwidth connection with the remaining client apparatuses. For example, the client apparatus(es) containing the stored event may include a server that has a connection to a plurality of televisions via a cable network, i.e. WEBTV. Similar functionality may be achieved via a broadcast medium. The present

embodiment is thus flexible by having an ability to host user events and cooperative events.

In one embodiment, the event includes a video and audio presentation such as movie, a concert, and/or a theatrical event. It should be noted, however, that the event may included any recording capable of being played back for entertainment, education, informative or other similar purposes.

In use, the client apparatuses and a host computer are adapted to be connected to a network. Such network may include a wide, local or any other type of communication network. For example, a wide area network such as the Internet may be employed which operates using TCP/IP or IPX protocols.

In operation **202**, information is transmitted from the host computer to the appropriate client apparatuses utilizing the network. This information allows for the simultaneous and synchronous playback of the event on each of the client apparatuses. In one embodiment, the information may also include a start time when the playback of the event is to begin on each of the client apparatuses. Further, an ending time may be included when the playback of the event is to end on each of the client apparatuses. Still yet, "play" command information may be sent to the client apparatuses at the start time. As an option, input may be received from the user, and used to alter the playback of the event. The host server, or synchronization server, can also control various streams of a variant rate and different hardware associated with those streams.

The present embodiment thus has the ability to synchronize video playback for one or multiple (thousands) users from one or multiple physical locations, and to synchronize with external video, audio and/or data streams.

Users of the present embodiment are at multiple physical locations and host servers may also be at different locations. The present embodiment is thus a scalable system which is capable of servicing an unlimited number of users. Since the content is local to the user machine, no high network bandwidth is required.

History Download Capabilities

Figure **3** illustrates a flowchart delineating a method for storing synchronization information for subsequent playback of an event. Initially, in

operation **300**, an event is stored in memory on at least one of the client apparatuses, as set forth earlier. These client apparatuses are adapted to be connected to a network along with a host computer during use.

In operation **302**, information is stored on the host computer(s) for allowing the simultaneous playback of the event on each of the client apparatuses. In one embodiment, the information may include a history and data associated with the synchronous playback. In particular, the history may include any overlaid material(as will be described hereinafter in greater detail), any specific commands affecting the playback of the information, or any other type of general information, i.e. start time, end time, etc.

In operation **304**, the information may be downloaded utilizing the network at any time after the synchronous playback of the event. Such downloaded information may then be used for playback after the simultaneous playback of the event. As such, the present embodiment has the ability to allow users to download a history and data associated with a particular synchronization event and play it later.

Overlay Synchronization

Figure **4** illustrates a flowchart setting forth a method for providing overlays during a synchronized event on a plurality of client apparatuses or any other source. First, in operation **400**, a plurality of client apparatuses are connected via a network. In operation **402**, an event may be simultaneously played back on the client apparatuses utilizing the network, as set forth earlier.

During the playback of the event, visual and/or audio material may also be overlaid on the event based on input received from at least one of the client apparatuses. See operation **404**. This may be accomplished by transmitting the overlay material from one of the client apparatuses to the host computer or any other server, and multicasting the same to the remaining client apparatuses.

As an option, the overlay material may include annotations on a display of the client apparatus. For example, the overlay material may include sketches which are inputted by way of a stylus-based input screen or a keyboard or the like, along with a voiceover inputted by way of a microphone or voice synthesizer. Such capability may also be quite valuable in an educational environment.

In one embodiment, the overlay material may also be displayed on each of the client apparatuses utilizing the network. This allows each of the users to experience the overlay in real-time during the simultaneous playback of the event. As an option, the user inputting the overlay material may select which users may experience the overlay material. The client apparatus that provided the overlay material may also be identified to the users experiencing the overlay material.

It should be noted that various bi-directional communication may be enabled for allowing data to travel to and from the server. For instance, the playback of the event on the client apparatuses may be altered in any feasible way based on input from a user.

Late Synchronization

Figure **5** illustrates a flow diagram for delayed synchronization of an event on a plurality of client apparatuses. First, in operation **500**, a plurality of client apparatuses are connected via a network and an event is stored in memory on the client apparatuses. The event is then simultaneously played back on the client apparatuses utilizing the network, as set forth earlier. Note operation **502**.

During the simultaneous playback, a request may be received from one of the client apparatuses for that particular to be included in the synchronized event, as set forth in operation **504**. This request may be received after the synchronized event has already begun while it is still playing. Further, the request may be submitted via a site on a network, i.e. website.

In response to the request, information is transmitted in operation **506** to the requesting client apparatus utilizing the network. This information is adapted for identifying a location in the memory where the event is currently being played back. This allows the simultaneous playback of the event on the requesting client apparatus.

The end users are thus able to come in at a later time and to be synchronized with the event. Targeted synchronization and various filters criteria can be applied to target different audiences. Also language and cultural differences can be taken into account. Still yet, the present embodiment may be adapted to address users on different hardware platforms (MAC, PC, set-top boxes). This may be accomplished

by identifying the user using a cookie, a user profile which is identified by way of a log in, or a Burn Cut Area (BCA) of the disc.

An example setting forth details relating to identifying DVDs will now be set forth. First, a content owner (such as studio) requests use of the BCA on their DVDs. Based on request, the replicator (examples include WAMO, Panasonic, Nimbus, Technicolor, Pioneer, Crest) adds unique BCA number to every DVD. Adding BCA number to each DVD requires a special (YAG) laser. This may be the very last step in the manufacturing process. The BCA numbers for a specific DVD must then be entered into InterActual's BCA database. Information to track includes: DVD title, i.e. "Lost in Space"; BCA #/range, i.e. 12345687890; and Shipping Packaging/Tracking Container, i.e. Box 52221 to Hollywood Video.

After the BCA number is added to the DVDs, the DVDs are packaging/boxed for distribution to either the Distributor or the Retailer. It should be noted that many companies take multiple forms, so the replicator and distributor may be one in the same. Also, some retailers are large/important enough to get shipments directly from replicator. The way in which the DVDs are packaging/shipped is very important because one must track the BCA numbers to actual shipping containers (box, etc.). Therefore tracking information must also be added to the BCA database.

If packaged DVDs are then sent to distributor, the distributor also has mechanisms, i.e. scanners, input device, and monitoring devices, in place for tracking based on their distribution. For example, Deluxe may receive a "package" of 100,000 copies of "Lost in Space." However, the distributor ships 10,000 to Retailer A and 5,000 to Retailer B. The distributor should be able to "input" retailer A and B's distribution information into the system. Ideally, this becomes a seamless/automated process.

Once the DVDs reach the retailer (either from the replicator or distributor), then DVDs may be further divided and distributed to local stores/outlets. In such a situation, the retailer should be able to automatically "track" distribution of these DVDs through to their stores. Over time, all three entitities (replicator, distributor, and retailer) are able to add tracking information to BCA database. Due to complexity and dependencies on existing business systems, the retail tracking concept will be rolled out in phases: replicator first most likely with key retail

accounts. The distributors will be brought in. Retailers will then begin to embrace the ability to track based on local outlet/store.

By the foregoing design, easy deployment is thus afforded and minimal hardware is required to allow the synchronization of content without significant capital investments and with a very efficient control mechanism. The content delivery does not rely on high network bandwidth and is independent from the synchronization.

Internet Server Application Program Interface (ISAPI) extensions will be used on the server. ISAPI extensions provide a mechanism to maintain a temporary or permanent connection with the users. These connections allow the Synchronization Server to process request and to send the appropriate DVD commands. The permanent connections are known as "Keep Alive" connections. ISAPI extension can also be used as an HTTP interface to a more traditional server, with all data returned as text.

On the client side the approach is to use, but not limited to Java 1.1 applets, to initiate event start-up for the Synchronization server. The advantage of using Java 1.1 applets is to achieve platform independence for existing and future Java-enabled devices. JavaScript will be used to provide user interface navigation by "wrapping" the applet.

An ISAPI (Internet Server Application Program Interface) is a set of Windows program calls that let one write a Web server application that will run faster than a Common Gateway Interface (CGI) application. A disadvantage of a CGI application (or "executable file," as it is sometimes called) is that each time it is run, it runs as a separate process with its own address space, resulting in extra instructions that have to be performed, especially if many instances of it are running on behalf of users. Using ISAPI, you create a Dynamic Link Library (DLL) application file that can run as part of the Hypertext Transport Protocol (HTTP) application's process and address space. The DLL files are loaded into the computer when HTTP is started and remain there as long as they are needed; they don't have to be located and read into storage as frequently as a CGI application.

Existing CGI applications can be converted into ISAPI application DLLs without having to rewrite their logic. However, they do need to be written to be thread-safe so that a single instance of the DLL can serve multiple users.

A special kind of ISAPI DLL is called an ISAPI filter, which can be designated to receive control for every HTTP request. One can create an ISAPI filter for encryption or decryption, for logging, for request screening, or for other purposes.

One can write ISAPI server extension DLLs (ISAs) that can be loaded and called by the HTTP server. Users can fill out forms and click a submit button to send data to a Web server and invoke an ISA, which can process the information to provide custom content or store it in a database. Web server extensions can use information in a database to build Web pages dynamically, and then send them to the client computers to be displayed. An application can add other custom functionality and provide data to the client using HTTP and HTML.

One can write an ISAPI filter. The filter is also a DLL that runs on an ISAPI-enabled HTTP server. The filter registers for notification of events such as logging on or URL mapping. When the selected events occur, the filter is called, and one can monitor and change the data (on its way from the server to the client or vice versa). ISAPI filters can be used to provide custom encryption or compression schemes, or additional authentication methods.

Both server extensions and filters run in the process space of the Web server, providing an efficient way to extend the server's capabilities.

Overall Component Design

The various functional components of the software associated with the present embodiment will now be set forth. Such components include a Java/JavaScript Component, Synchronizer Component, LayerImpl Component, Business Layer Component, Configuration Manager Component, and DBConnect Component.

Java/JavaScript Component

Figure **6** illustrates a flow diagram for providing information on a synchronized event on a plurality of client apparatuses in accordance with one embodiment of the present embodiment. First, in operation **600**, a plurality of client

apparatuses are connected via a network, as set forth earlier. Next, an application program is embedded on a site on the network in operation **602**. Such application program may take the form of a JAVA applet, and the site may include a website on the Internet.

In use, information is requested from a server on the network utilizing the application program. See operation **604**. Such information relates to an event to be played back simultaneously on the client apparatuses and may include general information such as a start and stop time of the event, or more specific information about the event itself.

In response to such request, a script is received for displaying the information. Note operation **606**. The script may take any form such as Perl, REXX (on IBM mainframes), and Tcl/Tk, and preferably includes a JAVAscript.

In one variation of the present embodiment, the JAVA applet may be further adapted to send a request to retrieve command information from the server for use with a playback device of one of the client apparatuses. The commands may be adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses. Further, the commands may include a start time when the playback of the event is to begin on each of the client apparatuses.

The JAVA applets and JAVAscript are used to communicate with the playback device of the client apparatuses. In one embodiment, the playback device includes a PCFriendly TM video player manufactured by Interactual ®.

The Java applet is embedded within a web page and uses HTTP protocol to communicate to the synchronization server. The applet could request event information from the server, and display it to the user via JavaScript. The applet could also send a "*BroadcastVideoEvent*" request to retrieve DVD commands that can be passed to the video component, as set forth hereinabove.

Synchronizer Component

Figure **7** illustrates a method for creating a synchronizer object in order to playback an event simultaneously on a plurality of client apparatuses. The synchronizer object is portion of the software that actually implements the

synchronization procedure. First, in operation **700**, a request is received utilizing a network for viewing an event. Next, the request is queued in memory in operation **702**.

In response to the request, in operation **704**, an object is created which is adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal. As an option, the activation signal may be provided using a clock of the client apparatus, or located at a different location, i.e. server. To accomplish this, the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

In operation **706**, the object is sent to one of the client apparatuses utilizing the network for being stored therein. In accordance with a primary aspect of the present embodiment, the object may be adapted to playback the event which is stored in memory of the client apparatus. This may be accomplished by activating a digital video disc (DVD) player.

In summary, when the Synchronizer component receives a "*BroadcastVideoEvent*" from the applet, it then places the request in the thread queue for processing. To process a request, the thread creates a "call back" object, if one does not exist for this event. The thread then adds the request to the "call back" object queue. This "call back" object will be invoked when it is time to play the DVD. The Synchronizer component creates a Call Back COM object, *LayerSink*. The Synchronizer component is also responsible for creating the *LayerFactory* interface which will be set forth hereinafter in greater detail.

LayerImpl Component

Figure **8** illustrates a flowchart for affording a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses. The present method ensures that critical information is tracked during the synchronization of the event. Such critical information not only ensures proper synchronization, but also enables various peripheral features.

First, in operation **800**, various values are determined including a current time, a start time when an event is to start, and a stop time when the event is to end.

Thereafter, a length of the event is calculated based on the start time and the stop time in operation **802**. As an option, the current time is determined by querying a clock of one of the client apparatuses.

If any portion of the length of the event takes place during a predetermined threshold period, a command is stored in memory in operation **804**. The command may be adapted to automatically begin playing back the event at the start time. In one embodiment, the threshold period includes the time the users can be queued before the event. As an option, chapter information may be stored in the memory if any portion of the length of the event takes place during the predetermined threshold period. This allows the command to automatically begin playing back the event at a predetermined chapter.

In operation **806**, a loop is created at the start time during which a lapsed time of the event is tracked. This information may be used for various tracking purposes to decide when to issue commands to the user. In another embodiment, a second loop may be created upon the beginning of a chapter during which information on a next chapter is retrieved.

The "call back" object (*LayerSink*) is thus responsible for creating and communicating with the *LayerImpl* component. The *LayerImpl* component acts as a scheduler, determining when to issue commands to the user.

LayerImpl will issue different DVD commands, based on the type of decoder the user has in their PC. *LayerImpl* will differentiate between the decoders by using the decoder information submitted from the client. The *LayerImpl* will pass the correct DVD command to the client, based on the decoder's capabilities. For example, if the decoder does not support the *TimePlay* event, then the server may send a *ChapterPlay* event and wait appropriately.

The following is an enumerated summary of the steps the component uses to determine when the users will receive the DVD commands:

1. Retrieves the current time, and the time the event starts and ends.
2. Calculates the length of the event.
3. If the event is within a threshold period (**i.e.** the time users can be queued before the event), then store the first DVD command in memory. Also, store the Chapter information in memory.
4. Create a loop that processes request until the event has completed.

5. In the loop, calculate the lapsed time of the event.
6. In the loop, retrieve the next chapter information.
7. Create another loop that will loop until time for the next chapter to be played.
8. When the next chapter is ready to play, send the command that was retrieved from the Chapter table.

Business Layer Component

Figure **9** is a flowchart delineating a method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses. This features is important since a host server may be synchronizing more than one event at once, or during overlapping times. Such events must therefore be distinguished.

First, in operation **900**, a plurality of events are stored in memory on a plurality of client apparatuses. Each of the events is assigned a unique identifier which is stored in the memory.

In operation **902**, the client apparatuses are adapted to be coupled to a host computer via a network, as set forth hereinabove. In operation **904**, the identifier of the event which is stored in the memory of the client apparatuses is then retrieved utilizing the network. Such identifier is subsequently compared with an identifier of a scheduled event, as set forth in operation **906**. If the comparison renders a match, the playback of the event is begun on the appropriate client apparatuses. Note operation **908**.

CbusinessLayer thus differentiates events by the disk and location ids, uploaded by the client to guarantee backwards compatibility. As set forth earlier, late arrivals can always re-sync with the event.

Configuration Manager Component

Figure **10** shows a flowchart delineating a technique for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event. The present technique is important since the playback devices of

the various client apparatuses may differ in make and model. Thus, different commands are required therefor.

In operation **1000**, a type of the playback devices of the client apparatuses is first identified. Such "type" may refer to a make, model, or any other distinguishing characteristic of the particular playback devices. A command associated with the identified type of the playback device is then looked up in a look-up table. Note operation **1002**. Such table may be located at the host server, or at any other location such as the client apparatuses.

Thereafter, in operation **1004**, the command is sent to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

This component is thus responsible for identifying what type of reference player is hosting the event. The reference player can be the database, which contains the DVD commands or a real time player. When the initial DVD is command is requested, the "Synchronizer" table is queried for the host type. From that point forward, the scheduler would know from whom to receive data.

DBConnect Component

This component is responsible for communicating with the Synchronizer tables, and for providing access methods for the retrieved data. All interaction from the tables is on a read-only basis. The *LayerImpl* component communicates with this component to retrieve DVD commands and event information.

Even though current implementation may be based on a Microsoft platform, hard dependencies on Microsoft or any other 3rd-party development tools may be avoided. To address such issues, the following considerations may be made throughout the code:

MFC specific code may be avoided. Instead, STL may be used. ATL and/or MFC code may be encapsulated into separate classes and portioned from the rest of the code. Class implementations may use aggregation pattern to delegate business logic to the portable classes. Database connection classes may be separated and the communication protocol may be separated with respect to portability to Oracle and other platforms.

Figures **11** and **12** illustrate the order of events among the various components of the present embodiment. In particular, Figure **11** illustrates the manner in which a layer factory is created. As shown, an event is first checked in a database server after which a business layer is created in a WEB server in a manner set forth hereinabove. The foregoing components are then created. Figure **12** illustrates the manner in which user requests are processed. As shown, communication is afforded with the video player on the client machine by means of JAVAscript and JAVA applets. The WEB server, in turn, communicates DVD commands to the video player via the JAVA applets, and also interfaces the database server via the various components thereof which were set forth hereinabove.

Alternate Embodiments

To support future enhancements, further components may be included with extendibility as the major objective. Various future enhancements of the product and how they will be addressed will now be set forth.

Hosted Real Time Players

While spirals may retrieve pre-recorded DVD commands from the database, alternate spirals may support a consumer as a host. The architecture may also support plug-in components. Alternate spirals may support the RealTimeConnector component, which accepts host user request and forwards them to the clients. The instant architecture supports the DBConnector which accepts events from the database.

Keep Alive Connections

Clients may maintain connections throughout the event. This allows the host to send a various number of commands to the client of the event. Although the spiral disconnects users once a PLAY command has been issued, the Synchronizer class (which will be set forth later) adds each connection to a Thread Pool. This pool of connections can be left open during the life of the event.

Logging Participants

Each request may be logged into the database to provide a reference for the future.

DVD Positioning

As an option, connections may be pooled to allow the synchronization server to direct consumer's machines to the certain locations throughout the entire event.

Synchronization events in alternate spirals may be defined as a combination of play from location event and the actual event. This way, one describes each event in the unambiguous way on the client side and synchronizes it with the server. For example, a situation may be considered where one fast forwards after a movie is played for 15 min and thereafter plays the scene in the movie. In such situation, one has to submit the information to the client player, indicating that it (player) has to start time play from 15 min into the movie and fast-forward to the certain location. A better way would be to analyze what is the next event after fast forwarding occurred and perform a combination for the play from location and next event. This design would require significant changes to the client infrastructure, including *video object*, *remoteagent* and *provider* and should be taken into consideration in any alternate client design.

Classes/Component Diagrams

Figures **13-16** illustrate various class/component diagrams. In particular, Figures **13-16** illustrate a Synchronizer Class Diagram **1300**, LayerImpl Class Diagram **1400**, Business Layer Class Diagram **1500**, and DBConnect Class Diagram **1600**, respectively.

Sequence Diagrams

Figure **17** illustrates a logical sequence diagram **1700**. As shown, when the server receives a user request, it analyzes the authentication information of the request (date/time, disc id, user id, and BCA number) and the appropriate synchronization event stored in the database. The database contains an *event start threshold* value measured in milliseconds. This threshold defines the amount of time prior to an event that a consumer is eligible to "connect" for the start of the event.

If the date/time of the user request lies within the event start threshold, the user is put into wait queue and receive the appropriate data when the time elapses. Note steps 1,2,3,5,6,7 of the Logical Sequence diagram. Otherwise, a message is sent informing the user when the event will occur. Note step 4 of the Logical Sequence diagram.

Server side collaboration diagram

Figure **18** illustrates a logical sequence diagram **1800** that shows server side collaboration. As shown, server ISAPI extension receives a BroadcastVideoEvents request. It calls *IA_BusinessServer* via *BeginProcess,* to retrieve configuration information. Configuration information contains a playback connector. Playback connector identifies whether the server will have to communicate with a reference player or will it perform playback from the database.

At step 6, ISAPI extension will call *IA_BusinessServer CompareTime* method and based on the results will send to the user a predefined web page indicating to retry later or return control to the web server, notifying it (web server) to keep the connection open. At this point connection is pooled and will be processed by the *IA_BusinessServer* at a time of the event.

Client Collaboration Diagram

Figure **19** illustrates a logical sequence diagram **1900** showing client side collaboration in accordance with one embodiment of the present embodiment.

Classes/Interfaces Definition

Definitions of one embodiment of the various classes associated with the software which implements the present embodiment will now be set forth.

Class Applet1

Purpose:

This is the class that implements the applet. The browser will use it to bootstrap our applet.

Responsibilities:

- Request a BroadCastVideo event and to gather event status information.

Collaborations:

BroadCastEvent, CITIEncrypt

Base class and implemented interfaces:

Javax.Applet

Public interface:

getChapter Returns the current chapter the reference player is playing.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getTitleInfo Returns the current title the reference player is playing
Return type: String

Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTime Returns the time the event is scheduled to start <SS:MM:HH:DD:MM:YYYY>
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeSec Returns the time the event starts in seconds.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeMinReturns the time the event starts in minutes.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getStartTimeHrReturns the time the event starts in Hours.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

GetStartTimeDay Returns the time the event starts in days.
Return type: String
Parameters: void
Pre-conditions: None.

Post-conditions: None.

GetStartTimeMnth Returns the time the event starts in months.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

GetStartTimeYr Returns the time the event starts in year.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

GetLenOfEvent Returns the length of the event.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

GetExpiredTime: Returns lapse time of the event.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTime: Returns the servers current time <SS:MM:HH:DD:MM:YYYY>.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeSec: Returns the servers current in seconds.

Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeMin: Returns the servers current in minutes.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeHr: Returns the servers current in hours.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeDay: Returns the servers current in day.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeMnth: Returns the servers current in month.
Return type: String
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getServerTimeYr: Returns the servers current in year.
Return type: String
Parameters: void
Pre-conditions: None.

Post-conditions: None.

startProc: Calls the ISAPIs "ServerInfo" method.
Return type: void
Parameters: String disk id, String location id
Pre-conditions: None.
Post-conditions: None.

msgEvent: Calls BroadCastEvent applet.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class BroadCastEvent

Purpose:

This is the class that invokes the Synchronizer.

Responsibilities:

- Sets the JavaScript with the command returned from the server.

Collaborations:

CITIEncrypt

Base class and implemented interfaces:

Java.Thread

Class CDBConnect

Purpose:

This is the class provides a public interface for components to request information from the DB tables.

Responsibilities:

- Opens the database and Synchronizer, Chapter_Disk tables.
- Queries the Synchronizer by the specified disk id and location id.
- Queries the Chapter_Disk by disk id.
- Provides the next chapter that is scheduled to play.
- Queries the Decoder_Capabilities table to determine if the requested player is time or chapter play.

Collaborations:

DBSyncSet
DBReferenceSet
CDBChapterSet
CDecoderCapabilities

Base class and implemented interfaces:

Public interface:

Get_NextChapter: Returns the next chapter to play.
Return type: String
Parameters: long time, long title, BSTR Chapter
Pre-conditions: None.
Post-conditions: None.

chkEvent: Checks if an event is scheduled for the disk and location id.

Return type: String
Parameters: long time, long title, BSTR Chapter
Pre-conditions: None.
Post-conditions: None.

get_initialDVDCommand: Returns the first DVD command to play.
Return type: String
Parameters: BSTR &
Pre-conditions: None.
Post-conditions: None.

get_nextDVDCommand: Returns the next DVD command to play.
Return type: String
Parameters: BSTR &
Pre-conditions: None.
Post-conditions: None.

decoderArray: Returns an array of decoder types.
Return type: String
Parameters: long **, long **
Pre-conditions: None.
Post-conditions: None.

Class CCConfigMgrImpl

Purpose:

This is the class provides a public interface for components to determine the type of reference player hosting the event.

Responsibilities:

- Opens the database and Synchronizer, Chapter_Disk tables.

- Queries the Synchronizer by the specified disk id and location id.
- Stores the reference player type.

Collaborations:

CConfigMgrRecSet

Base class and implemented interfaces:

Public interface:

get_hostType: Returns the reference player host type.
Return type: String
Parameters: short
Pre-conditions: None.
Post-conditions: None.

Class threadFunctor

Purpose:

This class provides a threading model that classes can use to derive.

Responsibilities:

- Calls the CreateEvent function, which opens a named or unnamed event objec.
- Calls _beginthread, which creates a thread begins execution of a routine at start_address. The routine at start_address must use the __cdecl calling convention and should have no return value. When the thread returns from that routine, it is terminated automatically.

- Calls the WaitForSingleObject function, which checks the current state of the specified object. If the object's state is nonsignaled, the calling thread enters an efficient wait state.
- Calls the ResetEvent function, which sets the state of the specified event object to nonsignaled.
- The state of an event object remains nonsignaled until it is explicitly set to signaled by the SetEvent or PulseEvent function.

Collaborations:

CConfigMgrRecSet

Base class and implemented interfaces:

Public interface:

start: Starts the thread.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

stop: Stops the thread. Calls CloseHandle for the thread and event.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class isapithread

Purpose:

This creates an ISAPI thread.

Responsibilities:

- Adds a request to a vector.
- Creates the sink object.
- Stores the request into sink object.
- Sends the time information to JavaScript.

Collaborations:

LayerSink
factorySink

Base class and implemented interfaces:

threadFunctor

Public interface:

addrequest: Adds the request to its vector.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

getBLayerInfo: Responsible for getting information about the event.
Return type: void
Parameters: std:string&,std::string&, ChttpServerContext*
Pre-conditions: None.
Post-conditions: None.

Class factorySink

Purpose:

Manages the layerSink and businessLayerProp objects.

Responsibilities:

- Stores a layerSink object.
- Returns the "businesssLayerProp" <Business Layer Properties>
- Creates the "businessLayerProp" <Business Layer structure>

Collaborations:

LayerSink
businessLayerProp

Base class and implemented interfaces:

Public interface:

construct: Stores a layerSink object..
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

notifyCreateLayer: Responsible for creating a "businessLayerProp".
Return type: void
Parameters: BSTR, BSTR, DATE, DATE, LONG
Pre-conditions: None.
Post-conditions: None.

Class layerSink

Purpose:

layerSink represents a sink interface and stores a queue of requests. It creates a connection point object.
This call back object, allows asynchronously processing.

Responsibilities:

- Acts as the client sink object.
- Sends the results to the user
- Creates the "BusinessLayer" and makes it a connection point object.
- Closes the users connection.
- Creates a Factory interface by calling "createFactory".
- Creates a connection point for the factory.
- Stores the LayerSink in the FactorySink object.
- Creates a connection point (call back) by calling AtlAdvise, between the connection point container and the client sink object. This allows the client to receive events.
- Calls the connectable objects "getServerLayer". This method fires an event to the clients sink object.
- Create a business layer,
- Store the request in its vector.
- Release the Sink Object (client)
- Calls AtlUnadvise to terminates the ability of the client to receive events.

Collaborations:

Base class and implemented interfaces:

Public interface:

construct: Creates a connection point.
Return type: void

Parameters: void
Pre-conditions: None.
Post-conditions: None.

addRequest: Adds the request to its vector.
Return type: void
Parameters: BSTR, BSTR, DATE, DATE, LONG
Pre-conditions: None.
Post-conditions: None.

createBusinessLayer: Creates a business layer. Create the connection point .
Return type: void
Parameters: businessLayerProp &
Pre-conditions: None.
Post-conditions: None.

updatetime: This call back function translates the time and sends the command to the user.
Return type: void
Parameters: long,long
Pre-conditions: None.
Post-conditions: None.

Class CBusinessLayer

Purpose:

Creates a layerthread object. This object is responsible for providing access methods, which provide event information.

Responsibilities:

- The "Synchronizers" createBusinessLayer method creates a class object from the "IBusinessLayer" interface. <The class object is part of the LayerImpl project>
- The BusinesLayers class object <m_ilayer> calls its "Initialize" method. <Note: m_ilayer is the connection point object. It identifies the "Sink Interface".
- It then calls the "Initialize" method of the connection point.
- The "Initialize" method then calls the "ChkValidEvent" method, which then creates a layerthread object.

Collaborations:

CBusinessLayer
layerthread

Base class and implemented interfaces:

Public interface:

Initialize: Calls the "ChkValidEvent" method which kicks of a layer thread.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class layerthread

Purpose:

This object acts as a scheduler, processing request from its queue.

Responsibilities:

- Send DVD commands to the user.
- "Syncs" up late comers to the events.

Collaborations:

CBusinessLayer
CDBConnect

Base class and implemented interfaces:

Public interface:

startThread: Processes requests from the queue
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

Class CLayerFactory

Purpose:

This object manages businesslayer objects. Business layer objects communicate with the reference player and notify the user which DVD command to play.

Responsibilities:

- Send DVD commands to the user.
- "Syncs" up late comers to the events.
- This object Implements the IID_LayerFactory interface.
- This COM object is the servers Connectable Point object.

- This server object supports connections to sink interfaces. These sink interfaces reside on the client side and are equivalent to the "call back" functions in Windows.

Collaborations:

CBusinessLayer
CDBConnect

Base class and implemented interfaces:

Public interface:

getServerLayer: "Fires" an event to create a business layer with the properties retrieved from the pipe object.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

put_set_layer: call the "CLayerFactoryImpl" add() method. Supplying the "businesslayer" object.
This will added to shared memory queue and written to a file.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

FinalConstruct: Calls the "CLayerFactoryImpl" FinalConstruct COM class object.
Return type: void
Parameters: void
Pre-conditions: None.
Post-conditions: None.

REMOTE CONTROL OF LOCAL CONTENT:

With reference to Fig. 20, the present embodiment provides a system and method for remote control of local content which enables the control of Video Playback from a remote server. Content stored on a medium 2002 such as a DVD is loaded onto a client device 2004. This hardware can be, for example, a computer, set top device such as is used to access WebTV, or some other device. The hardware device 2004 of the present embodiment has software 2006 in the form of a browser or presentation engine. In addition, the hardware 2004 has DVD Firmware or a Navigator 2008 in communication with the browser/presentation software 2006.

With continued reference to Fig. 20, a server 2010 delivers content to the hardware 2004 to be used in conjunction with the DVD 2002. This content can be in the form of ROM/HTML Content 2012 and/or DVD-Video Content 2014. Depending on the desired application, this content 2012 and 2014 enhances and/or allows a DVD experience 2016 provided by the DVD medium 2002.

With reference to FIG. 21, this control is performed by a transaction sever 2102 which sends video playback commands 2104 such as play stop, FF, Rewind, etc. It can also provide a locking/unlocking scheme which allows content on a local disk 2106 or website to be protected and accessible to particular users at prescribed points in time through a locking and unlocking process. This locking/unlocking technology could be broken down into two possible embodiments. For example, one such embodiment allows for unlocking local content that is on a local disk (i.e. DVD Disc) 2106 based on a user profile for example. In addition this content access could also expire or be accessible only during a particular time frame. Another possible embodiment allows for unlocking content on a website 2108 by requiring the user to have a DVD Disc in his computer's disc drive or set top box. (Therefore the user had to purchase the disk to get access to the on-line content).

This locking and unlocking is accomplished through the transaction server 2102, which validates the credentials of the user. These credentials 2110 are passed from the client 2112 (PC or set top box) and the server returns for example the unlock sequence 2114 to the client. In the case of DVD Video this unlock sequence can be in the form of General Purpose Registers Values (GPRM Bits) that unlock

the content. In the case of the website the client 2112 could pass the current disc 2106 in the drive's unique disc ID or BCA number and the transaction server 2102 allows a redirection to protected content after validation takes place. This unlocked content could be local or on the website 2108.

The advantages of remote control of a client's video device from a server is that the content can be protected. Since the information to use it is stored remotely, it can be easily maintained and upgraded, and allows introduction of new products without affecting the already shipped content (DVD Video). In addition, the control can be of a single client or many users. For example, with the ability to unlock content you can allow content to be accessible at a particular point in time thus allowing a special "event" or promotional time to occur and also allowing for various advertising/promotional models. The concept of expiring content is also useful, for example if an offer is only valid till the end of the year. The user is not burdened with viewing advertising or offers that they cannot participate in anyway after they have expired.

Another example is to reward users for purchasing particular products or even registering their products. The user can then be provided with additional content that is unlocked on the disc. In addition we can verify that people have the correct credentials before accessing content. To explain this further there may be website material that should only be accessible to customers who have purchased a particular DVD. The website may have additional information, games, or special items to be purchased and offered and this should only be made available to people who have purchased the DVD product. This may also span across multiple products for example if a user has purchased all of the available Lethal Weapon ™ titles it may be desirable to give additional content to that user for having purchased the series. Another embodiment of the embodiment follows the DIVX DVD model wherein a client is charged on a per usage basis for the content.

The present embodiment can also provide for remote navigation of content on a local server. For example navigation commands 2104 (Fig. 21) for left, right, up, down can be sent from the server to set General Purpose Registers (GPRMs) in the DVD Player that allowed content to be unlocked and viewed by users during the event. In addition, DVD navigation commands can be sent through streamed audio

with embedded triggers that send DVD navigation commands that call the video object in the web page.

In addition, With reference to Fig. 22, a synchronization server 2202 can be used to send commands to initiate video play in synchronization with several users 2112. Control can be of one or multiple clients in the form of PCs 2204 or set top players 2206. The remote navigation commands 2208 allow the server to tell the client what to do. The same set of commands can be sent to each of the clients – thus synchronizing the viewing experience. They could also be different, for example each user could be viewing a different DVD and therefore experiencing a different set of content. Based on the users profile 2210 they can also have access to different content. Given a geographical location or native language the control maybe tailored accordingly. The control of the video can be as simple as play, stop, fast forward, rewind, etc. or can include advanced features such as pan, zoom, rotate etc. The type of navigation/control can be divided into 3 aspects:

- **Commands**. Commands control the playback and search mechanisms of a DVD-Video disc. (These can be issued from the server or the client)
- **Properties**. Properties are used to query attributes of the DVD-Video and set certain configuration properties. (These can be queried from the server or the client). An example is to get the current state or title/chapter/time of the DVD Video). Another example is to tell the user whether or not he or she is on a menu.
- **Events**. Events are used to trigger notification of various playback conditions, such as time changes, title changes and UOP changes. Events are essential for scripting and synchronizing the video with other assets. (these are sent from the content or client back to the server. They can indicate that the video has stopped playing, you are on a menu item, or the number to angles available in the video has changed.

Embodiment in a web page

With reference to FIG. 23, an embodiment of this embodiment provides control of content through a web page 2302. Using a browser as the client interface

(Such as Internet Explorer) the user can browse to the page 2302 on-line that contains an Active-X Control that has an embedded video object 2304. The client sends identification information and/or requests 2305 to the web page over the Internet 2306. In response, the video object 2304 activates video navigation commands to play and/or unlock sequences 2308 required to play the video.

With reference to FIG. 24, in another embodiment of the embodiment the Video Object 2304 opens a secondary connection 2402 to the transaction server 2102 to receive commands 2404 from the server to execute.

In another form the browser interprets http commands for the control of video. The web page on the server is viewed by the client and when the user selects an item in the web page the HTTP link can be formatted with parameters for the browser to interpret directly for video playback.

For example http://iti_video? command=Play

This is interpreted by the browser since it has an iti_video in the url and then parses the parameters, which in this case is the command to play.

Examples of embedding a Video Object in a Web Page

DVD-Video can be embedded within a HTML page and control its layout. Computer operating systems can embed DVD-Video using currently available embedding techniques. By way of example, each of the major computer operating systems is provided below:

Operating System	**Example**
Windows	<object ID=" Video Object" CLASSID="clsid:A0739DE5-571F-11D2-A031-0060977F760C" BORDER="1" WIDTH=50% HEIGHT=60% > </object>
Apple/Macintosh	<embed ID=" Video Object" TYPE="application/x- Video Object -plugin"

	ALT=" Video Object Plug In" HIDDEN="TRUE" > </embed>
Linux	TBD
Others	TBD

After the DVD-Video object is embedded in the web page, it can be accessed using any style sheet, link, or scripting language. Values for the IDstring must begin with a letter (A-Z or a-z) and may be followed by any number of letters, digits, hyphens, and periods up to a maximum of 48.

Unlike computers, set-top boxes do not generally have a full-blown operating system and browser. Therefore, the capabilities within the browser are often more restricted. For embedding DVD-Video within these platforms using ITX, the "Video Object" ID must be integrated within the embedded browser as any other tag structure. With this approach, any embedded browser that encounters the "Video Object" tag, would automatically associate this identifier.

Unlocking Implementation:

Another embodiment of the embodiment provides a system and method for unlocking portions of DVD-Video based on certain criteria (date, profile, BCA, etc.). To control playback of video, the video can be "locked" so that the consumer must perform certain steps to access and play the video. The steps that trigger the unlock should be controlled by the content owner and can be based on date, consumer profile, BCA number or any other criteria, and should be controllable over the Internet from a remote server. (Although it is possible to store the unlock sequence locally as well). With reference to FIG. 25, a method 2502 is provided for controlling content. The method 2502 begins with a step 2504 wherein a user tries to play a portion of DVD-Video from application or web page. In a step 2506, video software initiates a secure connection to a transaction server that authenticates the user, and then passes the correct unlock sequence of events back to the video software. This video software is preferably stored locally on the user's computer or, more preferably, on the disk on which the video content is stored. In a step 2508,

the events are passed from the server, and then in a step 2510 are then passed by the video software directly to the underlying hardware or software DVD decoder, thereby bypassing any user knowledge of the events. This approach requires certain DVD-Video authoring requirements: (1) interleaving video and audio streams to prevent "back door" playback/access; (2) ability to populate GPRMs to create locking sequence.

There are two parts to the DVD-Video unlocking mechanism:

- The actual unlock process – performs actual unlocking of video. Without unlock process, consumer cannot access video. Therefore software on the remote server is required to unlock the video and other players will not support this feature.
- The protection process – the process protects against malicious consumers who try to bypass unlocking process. This process is not an actual locking process, but manipulates and distorts the video, thus rendering it non-viewable by consumers that try to bypass the unlock mechanism.

Unlock process:

The locking process is performed during the video authoring process of the DVD-Video. Each portion of video to be unlocked can be authored into a separate title, or title/chapter combination. (All references to locking video also apply to locking DVD-Audio)

The locking of the video utilizes General Parameter Registers (GPRMs), which are inherent in the DVD-Video specification. A DVD-Video title can be authored such that the GPRMs must be set to a specific value in order to allow an action to occur. In the case of our video unlocking scheme, the process of locking a video is ensuring that a video can only be played when the GPRMs are properly set. Then the DVD-Video is authored in such a way that remote server operator can programmatically (without consumer interaction) set the value of the GPRMs when certain conditions/transaction criteria has been met.

Fig. 26 illustrates a method 2602 for unlocking content. The steps required to create the "lock" or "key" for unlocking begin with a step 2604 of creating a title, the title being some form of content such as a movie video, training video or some other form of content. Thereafter, in a step 2606 hot spots are authored in the title (or use post command and jump to menu that contains hot spots). Then, in a step

2608, all of the hotspots are overlapped. In a step 2610 a sequence of events that trigger GPRMs is passed to the client. Once correct sequence of events are passed to client, resulting in populating the GPRMs properly, then, in a step 2612 the consumer is allowed to play the locked video.

Once correct GPRMs are populated, the appropriate post command is generated to jump the consumer to the locked title. Also note the GPRMs can be populated by either a direct call to an interface that allows setting the GPRM bits or through menu navigation commands such as left, right, up, and down.

It is also recommended to create a TIFF or animated graphic that displays when the DVD is placed into a traditional DVD consumer player. This TIFF or animated graphic can inform the user to place the DVD into a computer to access the special features and unlock the appropriate content. This information is displayed as soon as the First Play PGC is encountered.

Protection Process:

To avoid DVD playback solutions that violate the DVD-Video guidelines, additional precautions should be taken (these are not required, but recommended):

- Each section of video (title or chapter) to be locked should be interleaved with another portion of video (such as black, or with a second video to be locked. Since this requires the use of multi-angle content, the two sections of video must be of the exact length. This avoids certain decoders which ignore the DVD-Video layout and allow the consumer to play a VOB file directly. The first/default video stream (angle 1) should be the black, dummy video. The second video stream (angle 2) should be the actual content to be unlocked.
- Each section of video to be locked should contain two audio tracks: the first/default audio track should be noise, garbage or any audio that is NOT associated with the content to be unlocked. This is the default audio track that will be played if the consumer attempts to bypass the DVD-Video specification. The second audio track should be the actual audio track for the locked video.

This protection process is useful because interlaced/multi-angle video is formatted out sequentially. In other words, interlaced content is stored in the following manner: Seconds 0-2 video 1, seconds 0-2 video 2, seconds 3-5 video 1, seconds 3-5 video 2, etc.

Therefore if a consumer attempts to play the underlying VOB file directly, they will see video alternating every two seconds – which is very annoying. Additionally, if a VOB file is played using this approach, the default audio stream will play, which as defined above, will be garbage.

If utilizing these protection processes: after unlock process has completed successfully, the DVD-Video should be authored to play the appropriate title/chapter combination, as well as defaulting to the second video stream (angle 2) and second audio stream (audio track 2). Relate to BCA based on distribution channel allow access to content.

Control is on the Server side and it controls the client. Therefore server can give commands for content. The server can also create a game out of the navigation if they view certain clips in a certain order they will essentially be walking through a key setting scheme and thus unlocking further content. The Unlock information on a web site by requiring a DVD to be in the drive. The BCA number or Disk ID is passed to the website in the HTTP header and then this allows the content on the website to be accessed.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any other the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

CLAIMS

What is claimed is:

1. A method for synchronizing an event on a plurality of a client apparatuses, comprising the steps of:

(a) providing an event stored in a memory storage device;

(b) connecting the client apparatuses and a host computer to a network; and

(c) transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

2. A method as recited in claim 1, wherein the event includes a video and audio presentation.

3. A method as recited in claim 2, wherein the event includes at least one of a movie, a concert, and a theatrical event.

4. A method as recited in claim 1, wherein the network is a wide area network.

5. A method as recited in claim 1, wherein the information includes a start time when the playback of the event is to begin on each of the client apparatuses.

6. A method as recited in claim 1, wherein the information includes an ending time when the playback of the event is to end on each of the client apparatuses.

7. A method as recited in claim 1, wherein the memory storage device includes a digital video disc (DVD) player.

8. A method as recited in claim 7, wherein the information includes chapter information associated with the DVD.

9. A method as recited in claim 1, and further comprising the step of receiving input from the user, and altering the playback based on the input.

10. A computer program embodied on a computer readable medium for synchronizing an event on a plurality of a client apparatuses, comprising:

(a) a code segment for providing an event stored in a memory storage device;

(b) a code segment for connecting the client apparatuses and a host computer to a network; and

(c) a code segment for transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

11. A computer program as recited in claim 10, wherein the event includes a video and audio presentation.

12. A computer program as recited in claim 11, wherein the event includes at least one of a movie, a concert, and a theatrical event.

13. A computer program as recited in claim 10, wherein the network is a wide area network.

14. A computer program as recited in claim 10, wherein the information includes a start time when the playback of the event is to begin on each of the client apparatuses.

15. A computer program as recited in claim 10, wherein the information includes an ending time when the playback of the event is to end on each of the client apparatuses.

16. A computer program as recited in claim 10, wherein the memory storage device includes a digital video disc (DVD) player.

17. A computer program as recited in claim 16, wherein the information includes chapter information associated with the DVD.

18. A computer program as recited in claim 10, and further comprising a code segment for receiving input from the user, and altering the playback based on the input.

19. A system for synchronizing an event on a plurality of a client apparatuses, comprising:

(a) logic for providing an event stored in a memory storage device;

(b) logic for connecting the client apparatuses and a host computer to a network; and

(c) logic for transmitting information from the host computer to the memory storage device utilizing the network for allowing the simultaneous playback of the event on each of the client apparatuses.

20. A method for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising the steps of:

(a) providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

21. A method as recited in claim 20, wherein the event includes a video and audio presentation.

22. A method as recited in claim 20, wherein the information includes a history and data associated with the simultaneous playback.

23. A method as recited in claim 20, wherein the network is a wide area network.

24. A method as recited in claim 20, wherein the memory includes a digital video disc (DVD).

25. A method as recited in claim 24, wherein the information includes chapter information associated with the DVD.

26. A computer program embodied on a computer readable medium for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising:

(a) a code segment for providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) a code segment for storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) a code segment for allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

27. A computer program as recited in claim 26, wherein the event includes a video and audio presentation.

28. A computer program as recited in claim 26, wherein the information includes a history and data associated with the simultaneous playback.

29. A computer program as recited in claim 26, wherein the network is a wide area network.

30. A computer program as recited in claim 26, wherein the memory includes a digital video disc (DVD).

31. A computer program as recited in claim 26, wherein the information includes chapter information associated with the DVD.

32. A system for storing synchronization information for subsequent playback of an event on a plurality of client apparatuses, comprising:

(a) logic for providing an event stored in memory on at least one of the client apparatuses, wherein the client apparatuses and a host computer are adapted to be connected to a network;

(b) logic for storing information on the host computer for allowing the simultaneous playback of the event from the memory on each of the client apparatuses; and

(c) logic for allowing the information to be downloaded utilizing the network for playback after the simultaneous playback.

33. A system as recited in claim 32, wherein the event includes a video and audio presentation.

34. A system as recited in claim 32, wherein the information includes a history and data associated with the simultaneous playback.

35. A system as recited in claim 32, wherein the network is a wide area network.

36. A system as recited in claim 32, wherein the memory includes a digital video disc (DVD).

37. A system as recited in claim 32, wherein the information includes chapter information associated with the DVD.

38. A method for providing overlays during a synchronized event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network;

(b) simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) overlaying material during the playback of the event based on input received utilizing the network.

39. A method as recited in claim 38, wherein the overlay material includes audible material.

40. A method as recited in claim 38, wherein the overlay material includes visible material.

41. A method as recited in claim 40, wherein the overlay material includes annotations on a display of the client apparatus.

42. A method as recited in claim 40, and further comprising the step of displaying the overlay material on each of the client apparatuses utilizing the network.

43. A method as recited in claim 42, wherein the overlay material is displayed during the simultaneous playback of the event.

44. A method as recited in claim 42, and further comprising the step of identifying the client apparatus that provided the overlay material.

45. A method as recited in claim 38, wherein the event is stored on a digital video disc (DVD).

46. A method as recited in claim 38, and further comprising the step of receiving the input, and altering the playback based on the input.

47. A computer program embodied on a computer readable medium for providing overlays during a synchronized event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network;

(b) a code segment for simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) a code segment for overlaying material during the playback of the event based on input received utilizing the network.

48. A computer program as recited in claim 47, wherein the overlay material includes audible material.

49. A computer program as recited in claim 47, wherein the overlay material includes visible material.

50. A computer program as recited in claim 49, wherein the overlay material includes annotations on a display of the client apparatus.

51. A computer program as recited in claim 49, and further comprising a code segment for displaying the overlay material on each of the client apparatuses utilizing the network.

52. A computer program as recited in claim 51, wherein the overlay material is displayed during the simultaneous playback of the event.

53. A computer program as recited in claim 51, and further comprising the a code segment for identifying the client apparatus that provided the overlay material.

54. A computer program as recited in claim 47, wherein the event is stored on a digital video disc (DVD).

55. A computer program as recited in claim 47, and further comprising a code segment for receiving the input, and altering the playback based on the input.

56. A system for providing overlays during a synchronized event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network;

(b) logic for simultaneously playing back an event on the client apparatuses utilizing the network; and

(c) logic for overlaying material during the playback of the event based on input received utilizing the network.

57. A method for delayed synchronization of an event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) simultaneously playing back the event on the client apparatuses utilizing the network;

(c) receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

58. A method as recited in claim 57, wherein the request is received utilizing the network.

59. A method as recited in claim 57, wherein the event includes at least one of a movie, a concert, and a theatrical event.

60. A method as recited in claim 57, wherein the memory includes a digital video disc (DVD).

61. A method as recited in claim 60, wherein the information includes chapter information associated with the DVD.

62. A method as recited in claim 57, wherein the network is a wide area network.

63. A computer program embodied on a computer readable medium for delayed synchronization of an event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) a code segment for simultaneously playing back the event on the client apparatuses utilizing the network;

(c) a code segment for receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) a code segment for transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

64. A computer program as recited in claim 63, wherein the request is received utilizing the network.

65. A computer program as recited in claim 63, wherein the event includes at least one of a movie, a concert, and a theatrical event.

66. A computer program as recited in claim 63, wherein the memory includes a digital video disc (DVD).

67. A computer program as recited in claim 66, wherein the information includes chapter information associated with the DVD.

68. A computer program as recited in claim 63, wherein the network is a wide area network.

69. A system for delayed synchronization of an event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network, wherein an event is stored in memory on the client apparatuses;

(b) logic for simultaneously playing back the event on the client apparatuses utilizing the network;

(c) logic for receiving a request from one of the client apparatuses during the simultaneous playback to be included in the synchronized event; and

(d) logic for transmitting information to the requesting client apparatus utilizing the network for identifying a location in the memory where the event is currently being played back so as to allow the simultaneous playback of the event on the requesting client apparatus.

70. A system as recited in claim 69, wherein the request is received utilizing the network.

71. A system as recited in claim 69, wherein the event includes at least one of a movie, a concert, and a theatrical event.

72. A system as recited in claim 69, wherein the memory includes a digital video disc (DVD).

73. A system as recited in claim 72, wherein the information includes chapter information associated with the DVD.

74. A system as recited in claim 69, wherein the network is a wide area network.

75. A method for synchronizing an event on a plurality of client apparatuses, comprising the steps of:

(a) connecting a plurality of client apparatuses via a network;

(b) embedding an application program on a site on the network;

(c) requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) receiving a script for displaying the information.

76. A method as recited in claim 75, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

77. A method as recited in claim 76, wherein the playback device includes a digital video disc (DVD) player.

78. A method as recited in claim 76, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

79. A method as recited in claim 76, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

80. A method as recited in claim 75, wherein application program is a JAVA applet and the script is JAVAscript.

81. A computer program embodied on a computer readable medium for synchronizing an event on a plurality of client apparatuses, comprising:

(a) a code segment for connecting a plurality of client apparatuses via a network;

(b) a code segment for embedding an application program on a site on the network;

(c) a code segment for requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) a code segment for receiving a script for displaying the information.

82. A computer program as recited in claim 81, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

83. A computer program as recited in claim 82, wherein the playback device includes a digital video disc (DVD) player.

84. A computer program as recited in claim 82, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

85. A computer program as recited in claim 82, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

86. A computer program as recited in claim 81, wherein application program is a JAVA applet and the script is JAVAscript.

87. A system for synchronizing an event on a plurality of client apparatuses, comprising:

(a) logic for connecting a plurality of client apparatuses via a network;

(b) logic for embedding an application program on a site on the network;

(c) logic for requesting information from a server on the network utilizing the application program, wherein the information relates to an event to be played back simultaneously on the client apparatuses; and

(d) logic for receiving a script for displaying the information.

88. A system as recited in claim 87, wherein the application program is further adapted to send a request to retrieve commands from the server for use with a playback device of one of the client apparatuses.

89. A system as recited in claim 88, wherein the playback device includes a digital video disc (DVD) player.

90. A system as recited in claim 88, wherein the commands are adapted to playback the event on the playback device simultaneous with the playback of the event on the remaining client apparatuses.

91. A system as recited in claim 88, wherein the command includes a start time when the playback of the event is to begin on each of the client apparatuses.

92. A system as recited in claim 87, wherein application program is a JAVA applet and the script is JAVAscript.

93. A method for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising the steps of:

(a) receiving a request utilizing a network for viewing an event;

(b) queuing the request in memory;

(c) creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) sending the object to one of the client apparatuses utilizing the network for being stored therein.

94. A method as recited in claim 93, wherein the request is received via an application program embedded in a site on the network.

95. A method as recited in claim 94, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

96. A method as recited in claim 95, wherein the memory includes a digital video disc (DVD).

97. A method as recited in claim 93, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

98. A method as recited in claim 93, wherein the activation signal is provided using a clock of the client apparatus.

99. A computer program embodied on a computer readable medium for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising:

(a) a code segment for receiving a request utilizing a network for viewing an event;

(b) a code segment for queuing the request in memory;

(c) a code segment for creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) a code segment for sending the object to one of the client apparatuses utilizing the network for being stored therein.

100. A computer program as recited in claim 99, wherein the request is received via an application program embedded in a site on the network.

101. A computer program as recited in claim 100, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

102. A computer program as recited in claim 101, wherein the memory includes a digital video disc (DVD).

103. A computer program as recited in claim 99, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

104. A computer program as recited in claim 99, wherein the activation signal is provided using a clock of the client apparatus.

105. A system for creating a synchronizer object in order to playback an event simultaneously on a plurality of a client apparatuses, comprising:

(a) logic for receiving a request utilizing a network for viewing an event;

(b) logic for queuing the request in memory;

(c) logic for creating an object in response to the request, the object adapted to playback the event on a client apparatus simultaneous with the playback of the event on the remaining client apparatuses upon the receipt of an activation signal; and

(d) logic for sending the object to one of the client apparatuses utilizing the network for being stored therein.

106. A system as recited in claim 105, wherein the request is received via an application program embedded in a site on the network.

107. A system as recited in claim 106, wherein the object is adapted to playback the event which is stored in memory of the client apparatus.

108. A system as recited in claim 107, wherein the memory includes a digital video disc (DVD).

109. A system as recited in claim 105, wherein the object identifies a start time when the playback of the event is to begin on each of the client apparatuses.

110. A system as recited in claim 105, wherein the activation signal is provided using a clock of the client apparatus.

111. A method for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising the steps of:

(a) determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) calculating a length of the event based on the start time and the stop time;

(c) storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) creating a loop at the start time during which a lapsed time of the event is tracked.

112. A method as recited in claim 111, wherein the current time is determined by querying a clock of one of the client apparatuses.

113. A method as recited in claim 111, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

114. A method as recited in claim 111, and further comprising the step of storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

115. A method as recited in claim 111, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

116. A method as recited in claim 115, and further comprising the step of creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

117. A computer program embodied on a computer readable medium for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising:

(a) a code segment for determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) a code segment for calculating a length of the event based on the start time and the stop time;

(c) a code segment for storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) a code segment for creating a loop at the start time during which a lapsed time of the event is tracked.

118. A computer program as recited in claim 117, wherein the current time is determined by querying a clock of one of the client apparatuses.

119. A computer program as recited in claim 117, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

120. A computer program as recited in claim 117, and further comprising a code segment for storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

121. A computer program as recited in claim 117, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

122. A computer program as recited in claim 115, and further comprising a code segment for creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

123. A system for providing a scheduler object adapted to facilitate the playback of an event simultaneously on a plurality of networked client apparatuses, comprising:

(a) logic for determining a current time, a start time when an event is to start, and a stop time when the event is to end;

(b) logic for calculating a length of the event based on the start time and the stop time;

(c) logic for storing a command in memory if any portion of the length of the event takes place during a predetermined threshold period; and

(d) logic for creating a loop at the start time during which a lapsed time of the event is tracked.

124. A system as recited in claim 123, wherein the current time is determined by querying a clock of one of the client apparatuses.

125. A system as recited in claim 123, wherein the command is adapted to automatically begin playing back the event at the start time, and the event is stored in a memory of the client apparatus.

126. A system as recited in claim 123, and further comprising logic for storing chapter information in the memory if any portion of the length of the event takes place during a predetermined threshold period, and the memory includes a digital video disc (DVD).

127. A system as recited in claim 123, wherein chapter information is retrieved during the loop, and the memory includes a digital video disc (DVD).

128. A system as recited in claim 127, and further comprising logic for creating a second loop upon the beginning of a chapter during which information on a next chapter is retrieved.

129. A method for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising the steps of:

(a) providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) comparing the identifier with an identifier of a scheduled event; and

(d) beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

130. A method as recited in claim 129, wherein the event includes a video and audio presentation.

131. A method as recited in claim 129, wherein the event includes at least one of a movie, a concert, and a theatrical event.

132. A method as recited in claim 129, wherein the network is a wide area network.

133. A method as recited in claim 129, wherein the memory includes a digital video disc (DVD).

134. A computer program embodied on a computer readable medium for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising:

(a) a code segment for providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) a code segment for ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) a code segment for comparing the identifier with an identifier of a scheduled event; and

(d) a code segment for beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

135. A computer program as recited in claim 134, wherein the event includes a video and audio presentation.

136. A computer program as recited in claim 134, wherein the event includes at least one of a movie, a concert, and a theatrical event.

137. A computer program as recited in claim 134, wherein the network is a wide area network.

138. A computer program as recited in claim 134, wherein the memory includes a digital video disc (DVD).

139. A system for identifying a plurality of events which are played back simultaneously on a plurality of networked client apparatuses, comprising:

(a) logic for providing a plurality of events stored in memory on a plurality of client apparatuses, the events each having a unique identifier associated therewith and stored in the memory, wherein the client apparatuses are adapted to be coupled to a host computer via a network;

(b) logic for ascertaining the identifier of the event stored in the memory of the client apparatuses utilizing the network;

(c) logic for comparing the identifier with an identifier of a scheduled event; and

(d) logic for beginning the playback of the event on each of the client apparatuses if the comparison renders a match.

140. A system as recited in claim 139, wherein the event includes a video and audio presentation.

141. A system as recited in claim 139, wherein the event includes at least one of a movie, a concert, and a theatrical event.

142. A system as recited in claim 139, wherein the network is a wide area network.

143. A system as recited in claim 139, wherein the memory includes a digital video disc (DVD).

144. A method for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising the steps of:

(a) identifying a type of the playback device of each of the client apparatuses;

(b) looking up a command associated with the identified type of the playback device; and

(c) sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

145. A method as recited in claim 144, wherein the event includes a video and audio presentation.

146. A method as recited in claim 144, wherein the type of the playback device is identified utilizing the network.

147. A method as recited in claim 144, wherein the network is a wide area network.

148. A method as recited in claim 144, and further comprising the step of storing on the client apparatus an identifier of a host server that sent the command.

149. A method as recited in claim 144, wherein the memory includes a digital video disc (DVD).

150. A computer program embodied on a computer readable medium for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising:

(a) a code segment for identifying a type of the playback device of each of the client apparatuses;

(b) a code segment for looking up a command associated with the identified type of the playback device; and

(c) a code segment for sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

151. A computer program as recited in claim 150, wherein the event includes a video and audio presentation.

152. A computer program as recited in claim 150, wherein the type of the playback device is identified utilizing the network.

153. A computer program as recited in claim 150, wherein the network is a wide area network.

154. A computer program as recited in claim 150, and further comprising a code segment for storing on the client apparatus an identifier of a host server that sent the command.

155. A computer program as recited in claim 150, wherein the memory includes a digital video disc (DVD).

156. A system for identifying playback devices of a plurality of client apparatuses which are networked to simultaneously playback an event, comprising:

(a) logic for identifying a type of the playback device of each of the client apparatuses;

(b) logic for looking up a command associated with the identified type of the playback device; and

(c) logic for sending the command to the corresponding client apparatus for beginning the playback of the event simultaneously with the playback of the event on each of the remaining client apparatuses.

157. A system as recited in claim 156, wherein the event includes a video and audio presentation.

158. A system as recited in claim 156, wherein the type of the playback device is identified utilizing the network.

159. A system as recited in claim 156, wherein the network is a wide area network.

160. A system as recited in claim 156, and further comprising logic for storing on the client apparatus an identifier of a host server that sent the command.

161. A system as recited in claim 156, wherein the memory includes a digital video disc (DVD).

162. A method for remotely controlling digital content, the digital content being stored locally on a client device, the method comprising the steps of:

(a) coupling the client with a network to retrieve a client identification from the client device;

(b) generating a query, based upon the client identification, to determine whether the client should have access to the digital content; and

(c) sending an unlock command via the network to the client device, if the client should have access to the content stored on the client device, the command being operable to allow the client device to access and utilize the digital content stored locally on the client device.

163. A method for remotely controlling content as recited in claim 162 wherein the client identification includes an identification of a user operating the client device.

164. A method for remotely controlling digital content as recited in claim 162 wherein the client identification includes an identification of the client device.

165. A method for remotely controlling digital content as recited in claim 162, including the step of sending a navigation command via the network to the client device, the navigation command being operable to control navigation of the content stored locally on the client device.

166. A method for remotely controlling content as recited in claim 165, including the step of delivering additional content to the client device via the network, to supplement the content stored locally thereon.

167. A method for remotely controlling digital content as recited in claim 162 wherein the client device is a computer.

168. A method for remotely controlling digital content as recited in claim 162 wherein the client device is a set top box.

169. A method for remotely controlling digital content as recited in claim 162 wherein the locally stored content is embodied on a Digital Versatile Disc (DVD).

170. A method for remotely controlling content as recited in claim 162, including the step of initiating synchronous play of content on a plurality of client devices, each client device being connected to the network.

171. A method for remotely controlling digital content as recited in claim 162 wherein the network is the Internet.

172. A method for remotely controlling digital content as recited in claim 162 wherein the unlock command incrementally allows access to portions of the content stored on the client device based upon the determination of whether the client should have access to the content.

173. A computer program embodied on a computer readable medium for remotely controlling digital content which has been stored locally on a client device, the computer program comprising:

(a) a code segment that receives an input delivered over a network from the client device, the input including a client identifier;

(b) a code segment that queries the client input to determine, based upon the client identifier, whether the client should have access to the content;

(c) a code segment that unlocks the digital content stored on locally on the client device, allowing access to the content by the client device based upon the results of the query; and

(d) a code segment that delivers to the client device, via the network, the code segment that unlocks the locally stored content.

174. A computer program as recited in claim 173 wherein the client identifier includes an identification of the client device.

175. A computer program as recited in claim 173 wherein the client identifier includes an identification of a user operating the client device.

176. A computer program as recited in claim 175, including a code segment for supplying additional content to the client device to supplement the content stored locally thereon based upon the client identifier of the client device.

177. A computer program as recited in claim 173 wherein the digital content is a video stored on a Digital Versatile Disc (DVD).

178. A computer program as recited in claim 173 wherein the code segment for controlling the use of the content controls navigation of the content by the client device.

179. A computer program as recited in claim 178 further comprising a code segment for synchronizing play of the video on a plurality of client devices.

180. A system for remotely controlling digital content, the digital content being stored locally on a client device, the system comprising:

(a) a processor, remote from the client device;

(b) a memory, remote from the client device, that stores information under control of the processor;

(c) logic stored on the memory that retrieves and interprets input from the client device, the input being delivered over a network and including a client device identification;

(d) logic stored on the memory, that responds to the input from the client device to generate an unlock command, based upon the client device identification, to allow use of the locally stored digital content by the client device having the digital content stored thereon; and

(e) logic stored on the memory that delivers the unlock command over the network to the client device.

181. A system as recited in claim 180 wherein the client identification includes an identification of a user operating the client device.

182. A system as recited in claim 180 wherein the client identification includes an identification of the client device;

183. A system as recited in claim 180 including logic that controls navigation of the digital content stored locally on the client device.

184. A system as recited in claim 180, including logic that supplements the content stored locally on the client device with additional content delivered over the network.

185. A system as recited in claim 180 wherein the digital content on client device is initially embodied on a Digital Versatile Disk (DVD).

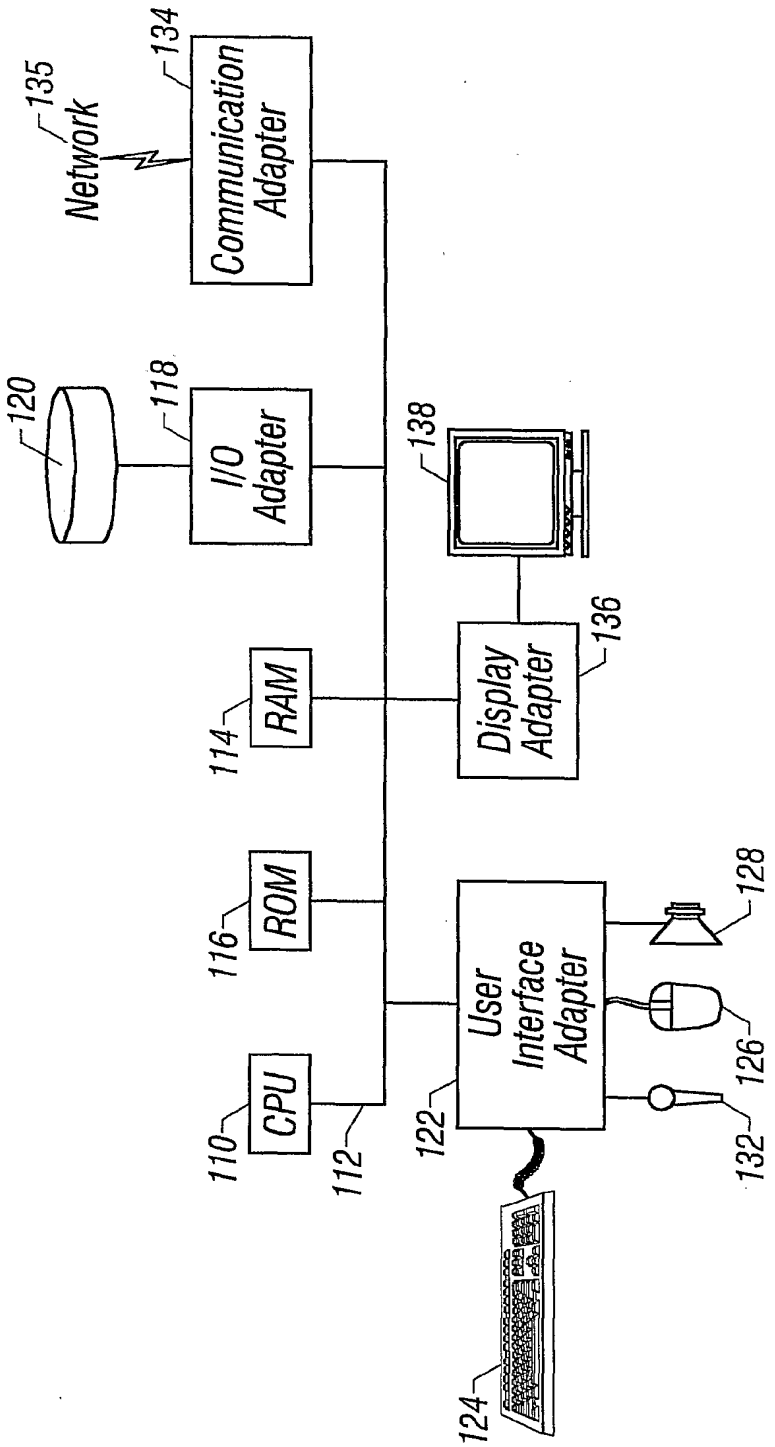

Network
135
Communication Adapter
134
120
I/O Adapter
118
138
RAM
114
Display Adapter
136
ROM
116
User Interface Adapter
128
CPU
110
112
122
126
132
124

FIG. 1

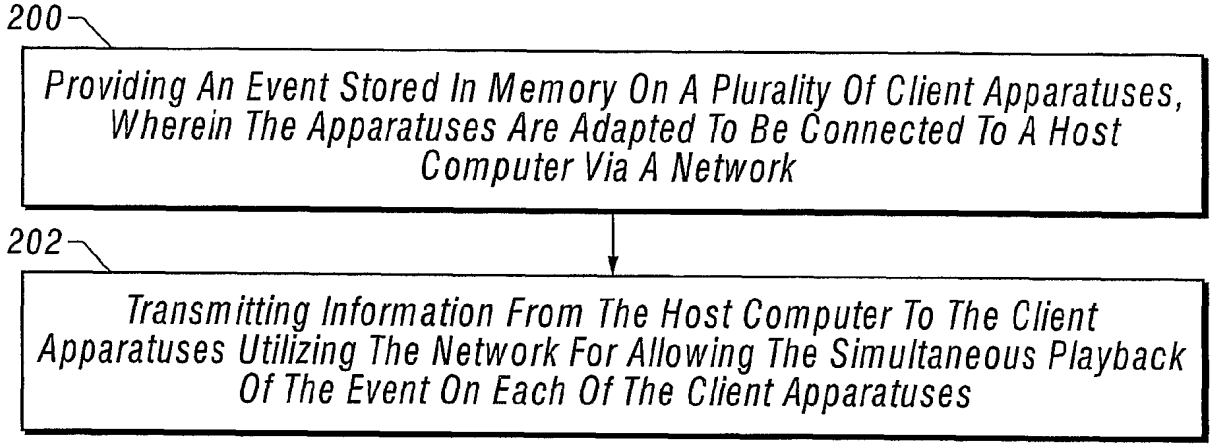

200
Providing An Event Stored In Memory On A Plurality Of Client Apparatuses, Wherein The Apparatuses Are Adapted To Be Connected To A Host Computer Via A Network
202
Transmitting Information From The Host Computer To The Client Apparatuses Utilizing The Network For Allowing The Simultaneous Playback Of The Event On Each Of The Client Apparatuses

FIG. 2

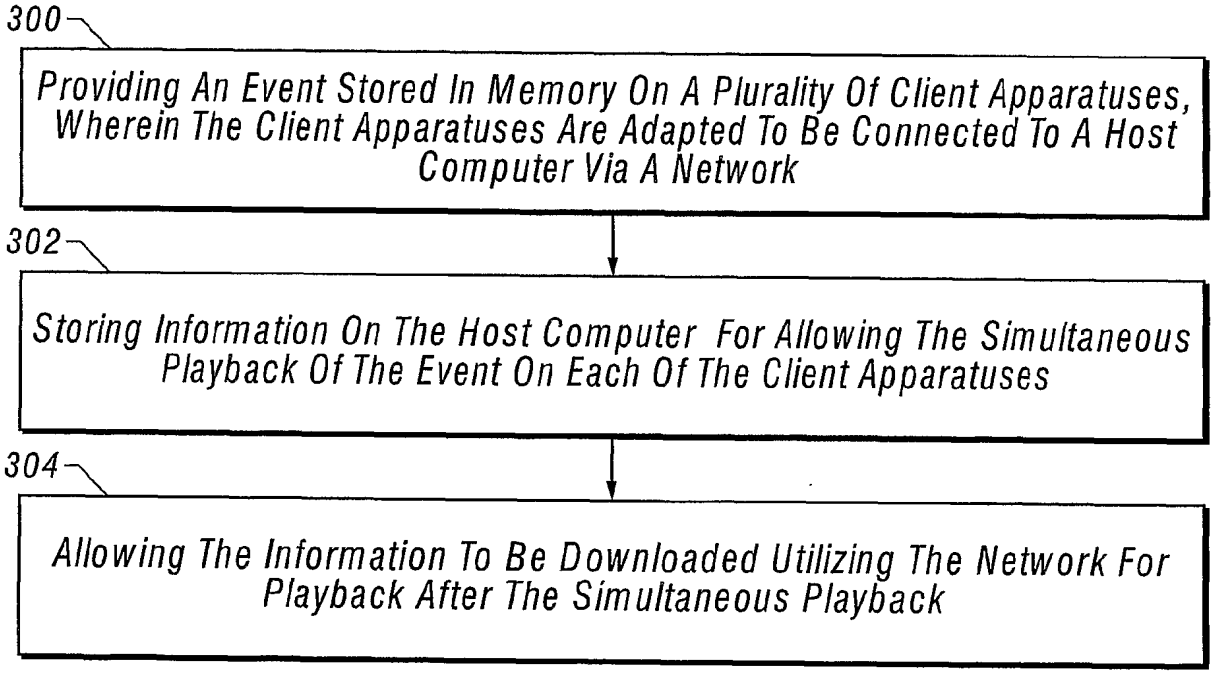

300
Providing An Event Stored In Memory On A Plurality Of Client Apparatuses, Wherein The Client Apparatuses Are Adapted To Be Connected To A Host Computer Via A Network
302
Storing Information On The Host Computer For Allowing The Simultaneous Playback Of The Event On Each Of The Client Apparatuses
304
Allowing The Information To Be Downloaded Utilizing The Network For Playback After The Simultaneous Playback

FIG. 3

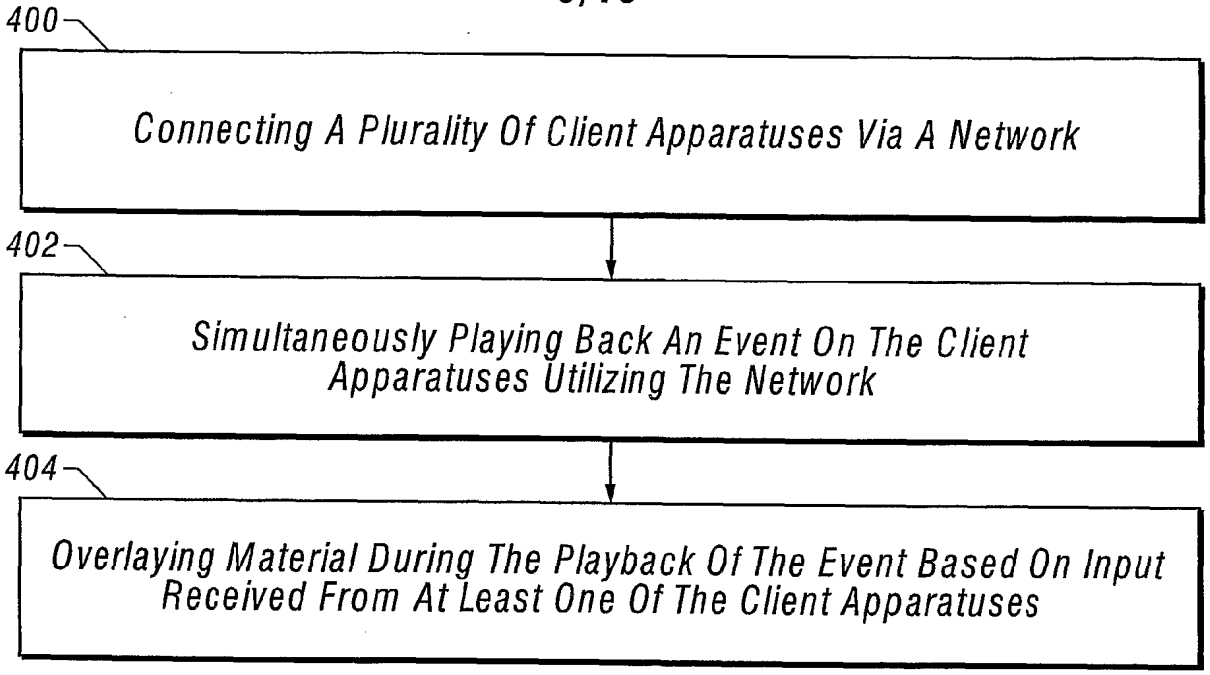

400
Connecting A Plurality Of Client Apparatuses Via A Network
402
Simultaneously Playing Back An Event On The Client Apparatuses Utilizing The Network
404
Overlaying Material During The Playback Of The Event Based On Input Received From At Least One Of The Client Apparatuses

FIG. 4

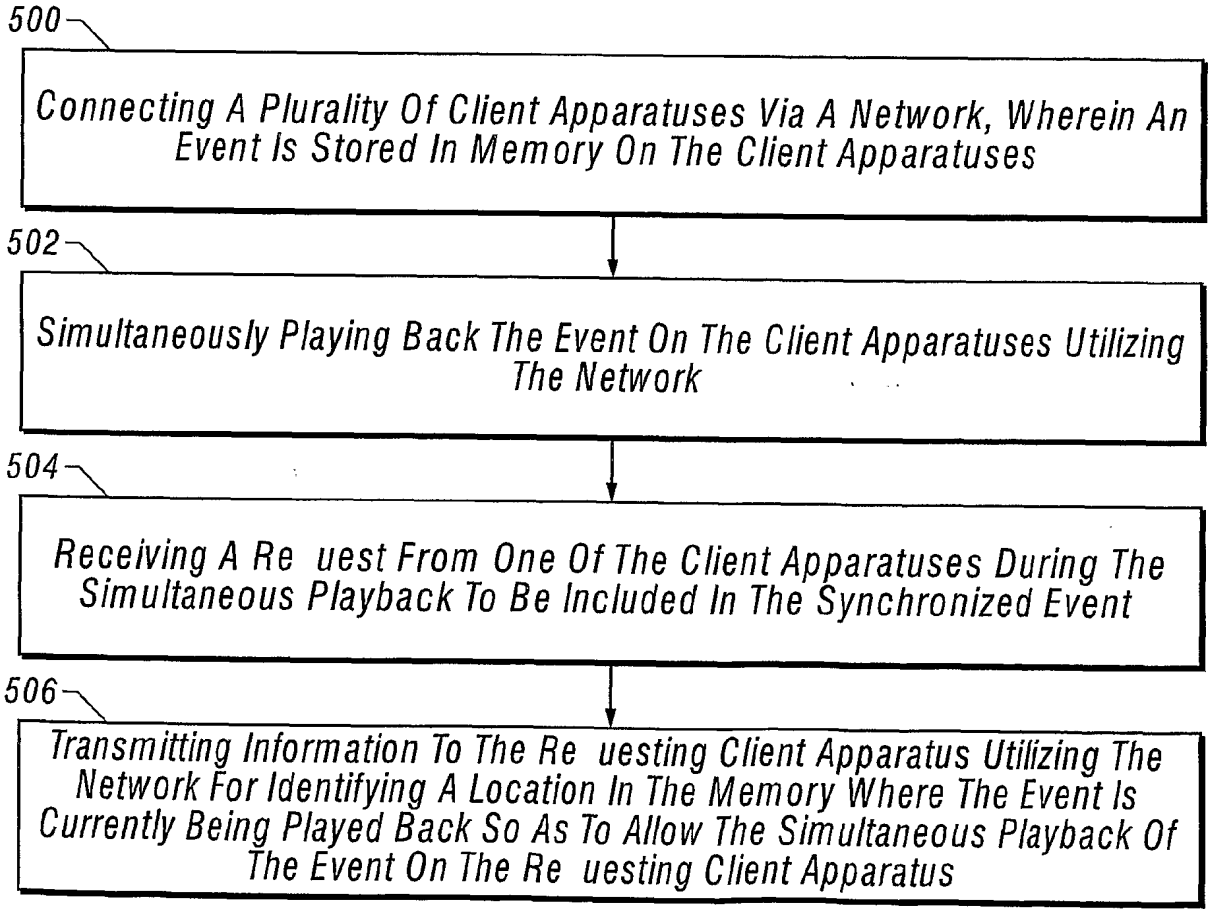

500
Connecting A Plurality Of Client Apparatuses Via A Network, Wherein An Event Is Stored In Memory On The Client Apparatuses
502
Simultaneously Playing Back The Event On The Client Apparatuses Utilizing The Network
504
Receiving A Re uest From One Of The Client Apparatuses During The Simultaneous Playback To Be Included In The Synchronized Event
506
Transmitting Information To The Re uesting Client Apparatus Utilizing The Network For Identifying A Location In The Memory Where The Event Is Currently Being Played Back So As To Allow The Simultaneous Playback Of The Event On The Re uesting Client Apparatus

FIG. 5

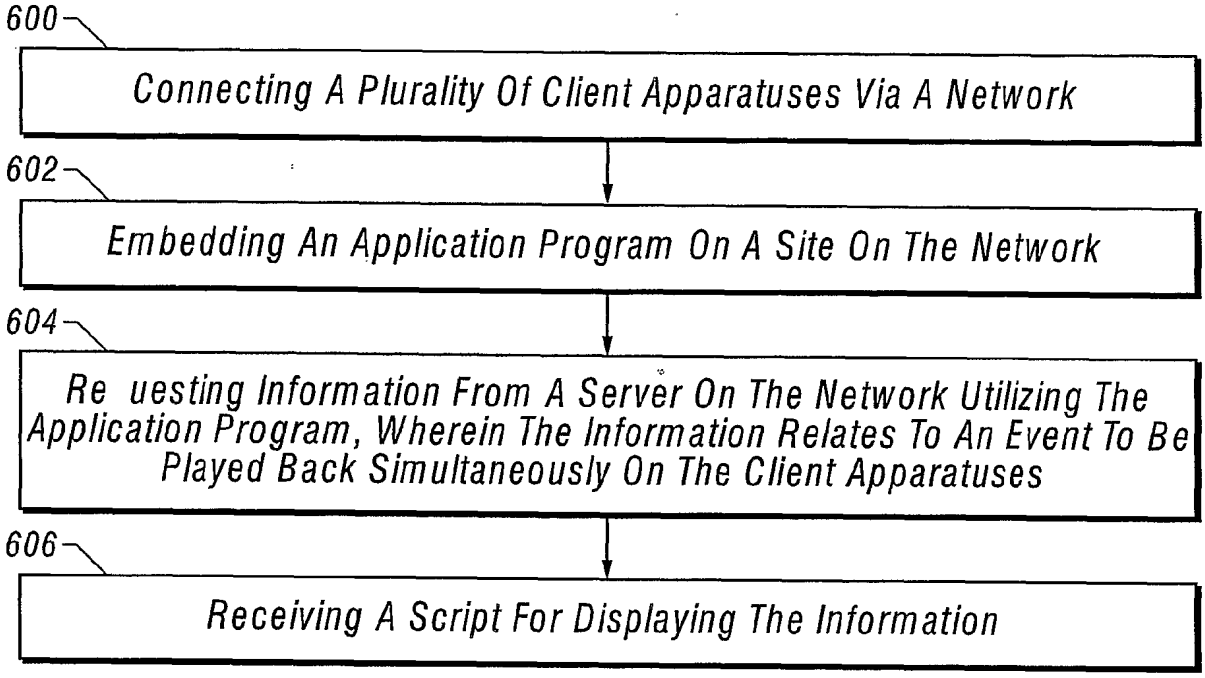
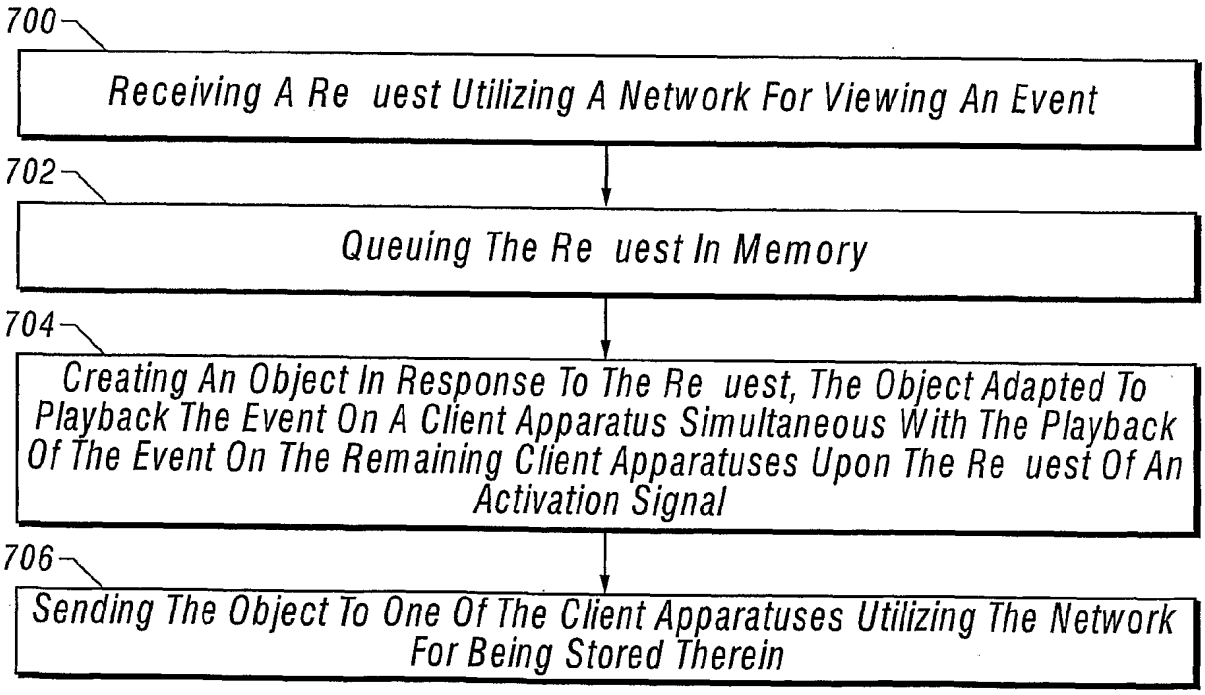

600
Connecting A Plurality Of Client Apparatuses Via A Network
602
Embedding An Application Program On A Site On The Network
604
Re uesting Information From A Server On The Network Utilizing The Application Program, Wherein The Information Relates To An Event To Be Played Back Simultaneously On The Client Apparatuses
606
Receiving A Script For Displaying The Information

FIG. 6


700
Receiving A Re uest Utilizing A Network For Viewing An Event
702
Queuing The Re uest In Memory
704
Creating An Object In Response To The Re uest, The Object Adapted To Playback The Event On A Client Apparatus Simultaneous With The Playback Of The Event On The Remaining Client Apparatuses Upon The Re uest Of An Activation Signal
706
Sending The Object To One Of The Client Apparatuses Utilizing The Network For Being Stored Therein

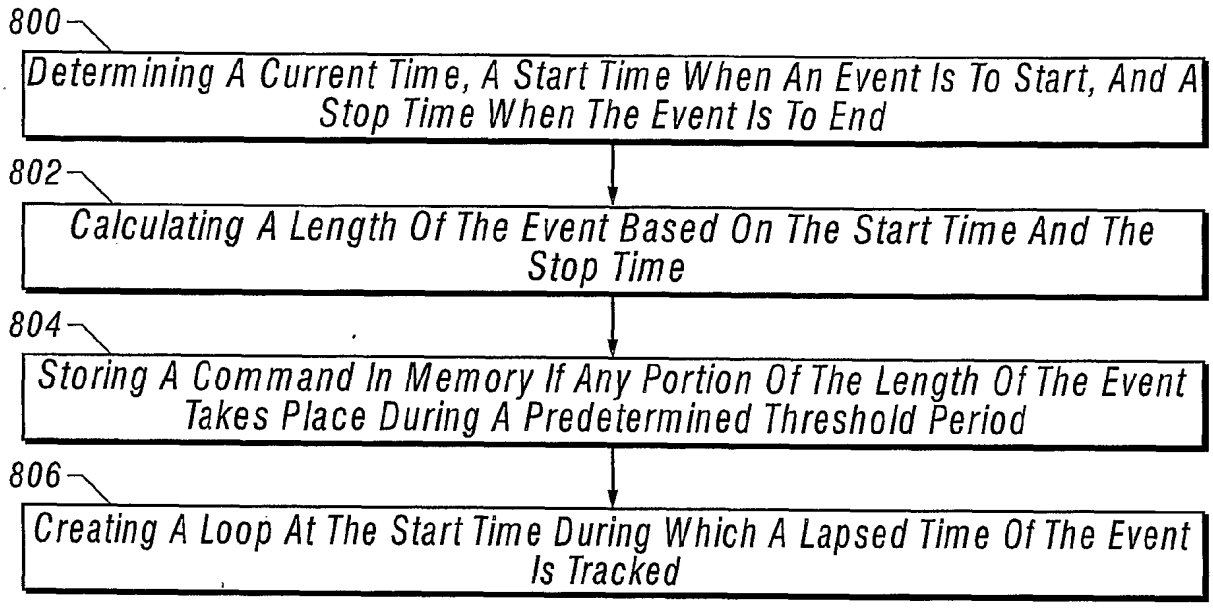

800
Determining A Current Time, A Start Time When An Event Is To Start, And A Stop Time When The Event Is To End
802
Calculating A Length Of The Event Based On The Start Time And The Stop Time
804
Storing A Command In Memory If Any Portion Of The Length Of The Event Takes Place During A Predetermined Threshold Period
806
Creating A Loop At The Start Time During Which A Lapsed Time Of The Event Is Tracked

FIG. 8

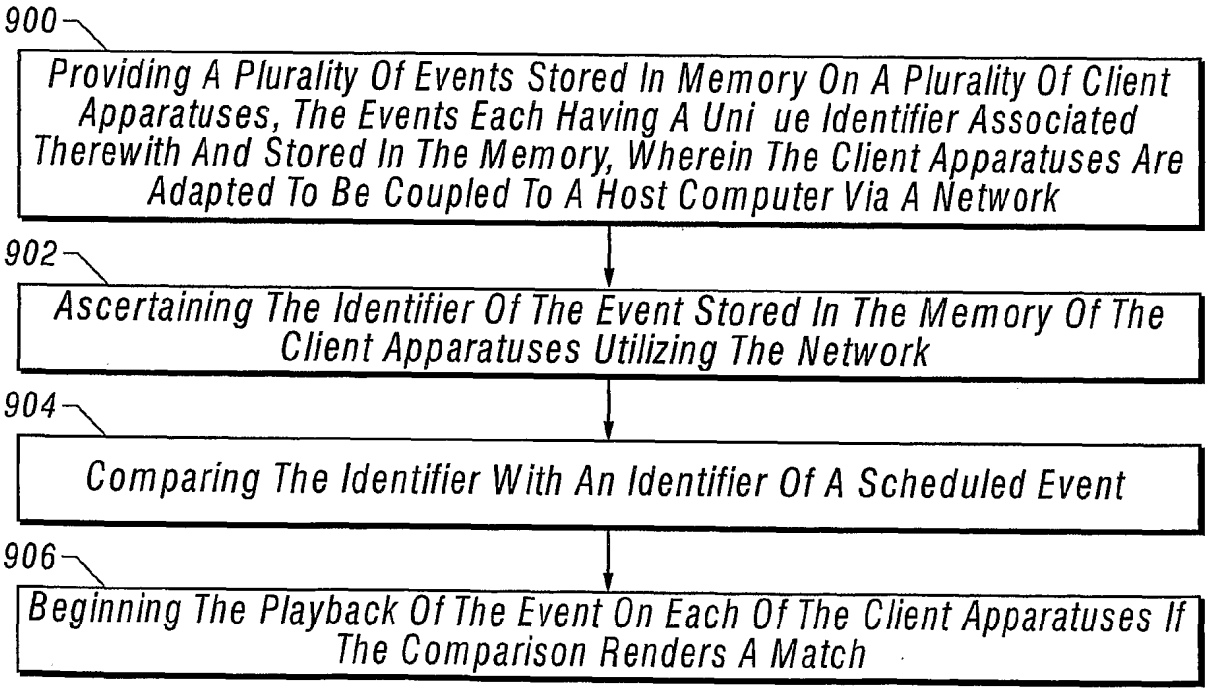

900
Providing A Plurality Of Events Stored In Memory On A Plurality Of Client Apparatuses, The Events Each Having A Uni ue Identifier Associated Therewith And Stored In The Memory, Wherein The Client Apparatuses Are Adapted To Be Coupled To A Host Computer Via A Network
902
Ascertaining The Identifier Of The Event Stored In The Memory Of The Client Apparatuses Utilizing The Network
904
Comparing The Identifier With An Identifier Of A Scheduled Event
906
Beginning The Playback Of The Event On Each Of The Client Apparatuses If The Comparison Renders A Match

FIG. 9

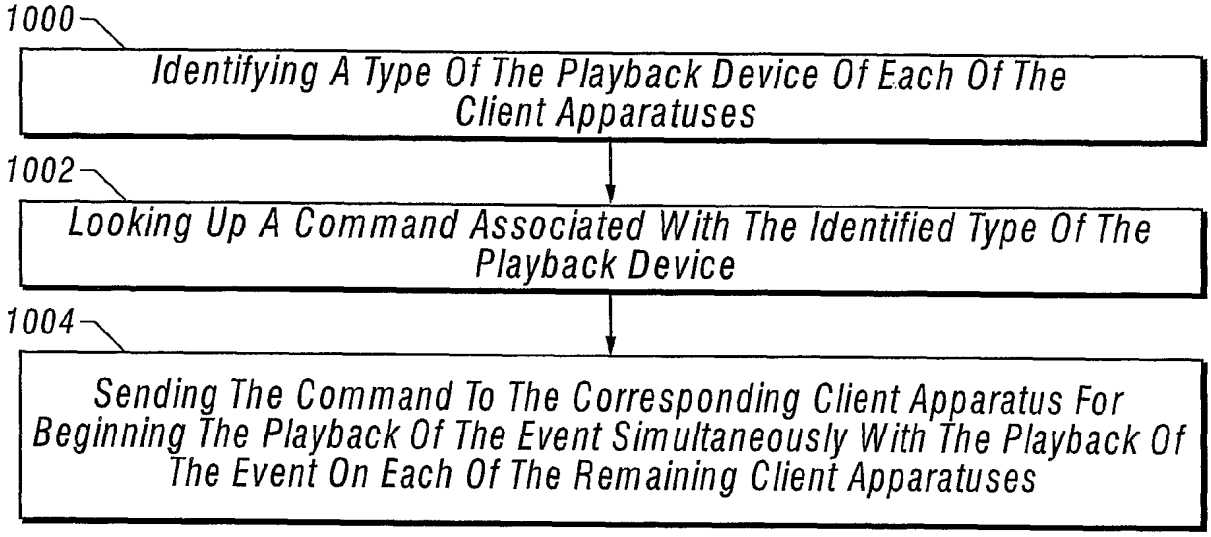

1000
Identifying A Type Of The Playback Device Of Each Of The Client Apparatuses
1002
Looking Up A Command Associated With The Identified Type Of The Playback Device
1004
Sending The Command To The Corresponding Client Apparatus For Beginning The Playback Of The Event Simultaneously With The Playback Of The Event On Each Of The Remaining Client Apparatuses

FIG. 10

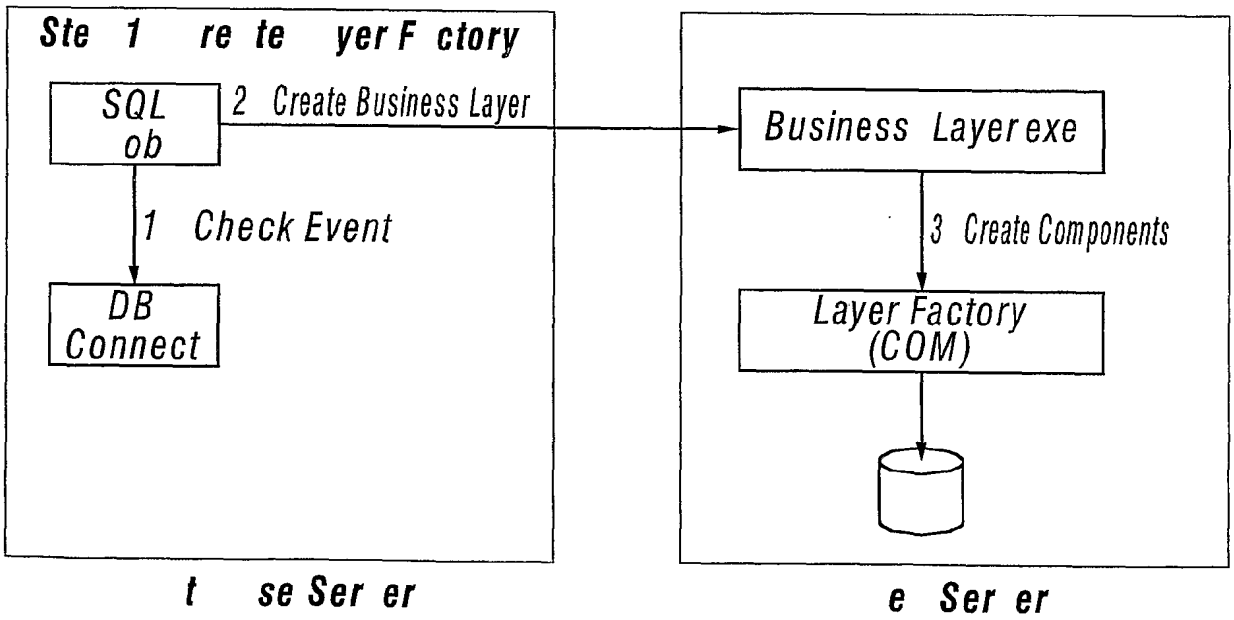

Ste 1 re te yer F ctory
SQL ob
2 Create Business Layer
1 Check Event
DB Connect
Business Layer exe
3 Create Components
Layer Factory (COM)
t se Ser er
e Ser er

FIG. 11

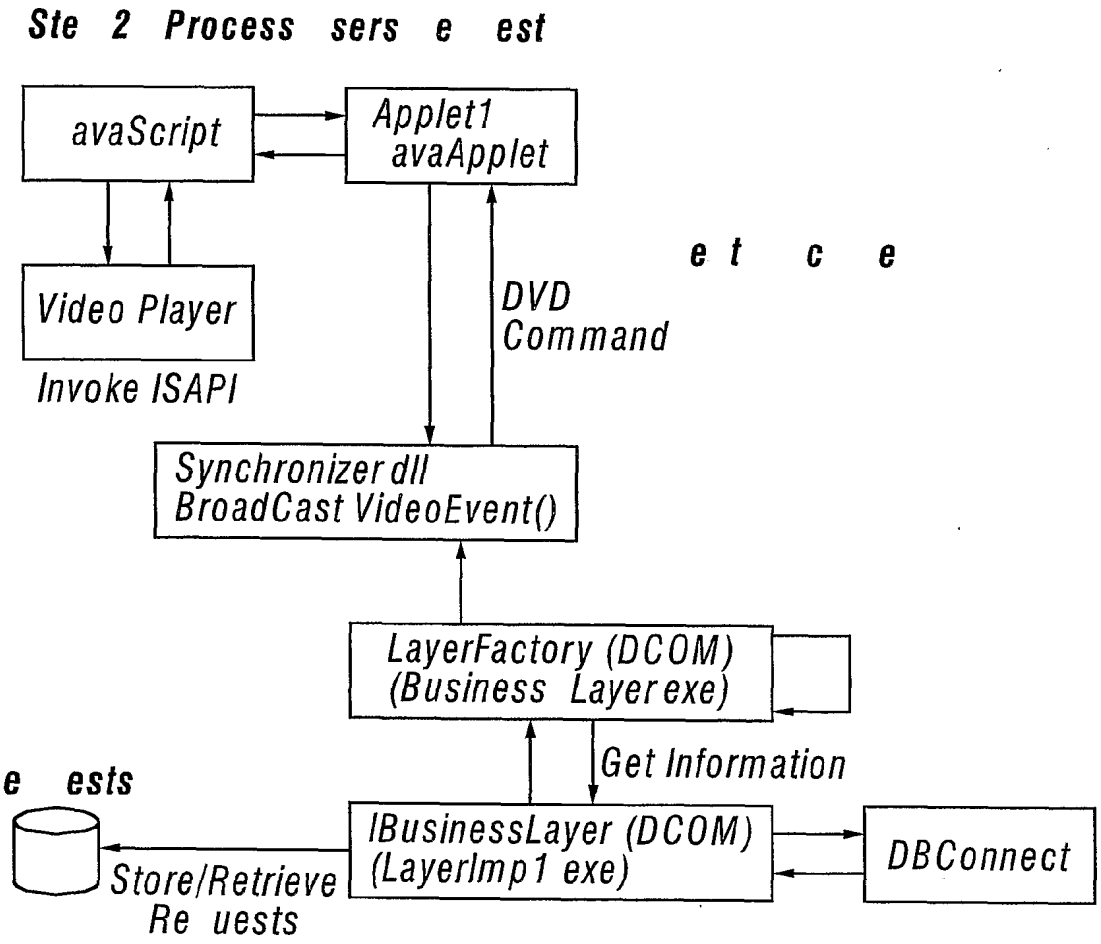

Ste 2 Process sers e est
avaScript
Applet1
avaApplet
e t c e
Video Player
DVD
Command
Invoke ISAPI
Synchronizer dll
BroadCast VideoEvent()
LayerFactory (DCOM)
(Business Layer exe)
Get Information
e ests
IBusinessLayer (DCOM)
(LayerImp1 exe)
DBConnect
Store/Retrieve
Re uests

FIG. 12

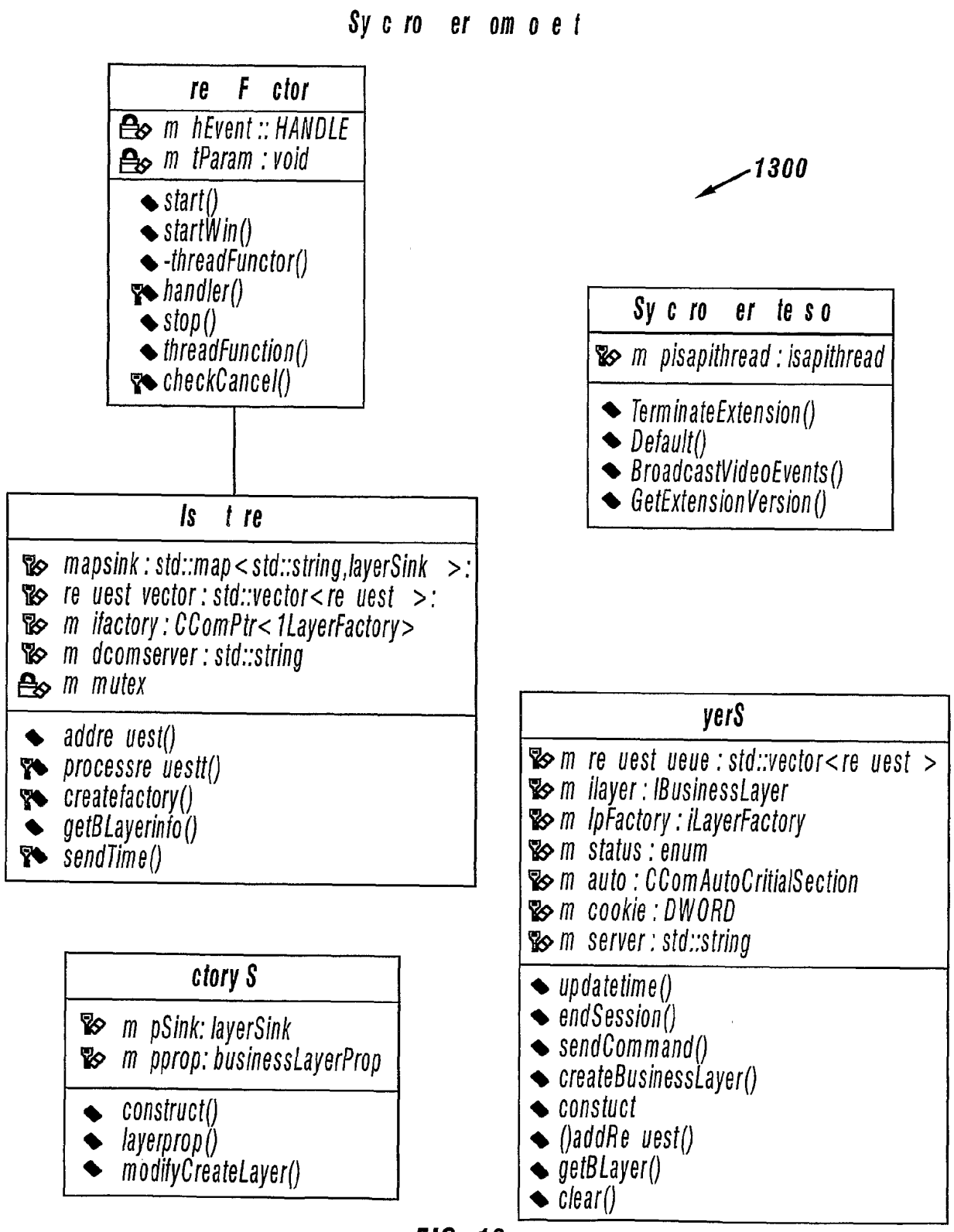

Sy c ro er om o e t
re F ctor
m hEvent :: HANDLE
m tParam : void
start()
startWin()
-threadFunctor()
handler()
stop()
threadFunction()
checkCancel()
1300
Sy c ro er te s o
m pisapithread : isapithread
TerminateExtension()
Default()
BroadcastVideoEvents()
GetExtensionVersion()
Is t re
mapsink : std::map<std::string,layerSink >:
re uest vector : std::vector<re uest >:
m ifactory : CComPtr< 1LayerFactory>
m dcomserver : std::string
m mutex
addre uest()
processre uestt()
createfactory()
getBLayerinfo()
sendTime()
yerS
m re uest ueue : std::vector<re uest >
m ilayer : IBusinessLayer
m lpFactory : iLayerFactory
m status : enum
m auto : CComAutoCritialSection
m cookie : DWORD
m server : std::string
updatetime()
endSession()
sendCommand()
createBusinessLayer()
constuct
()addRe uest()
getBLayer()
clear()
ctory S
m pSink: layerSink
m pprop: businessLayerProp
construct()
layerprop()
modifyCreateLayer()

FIG. 13

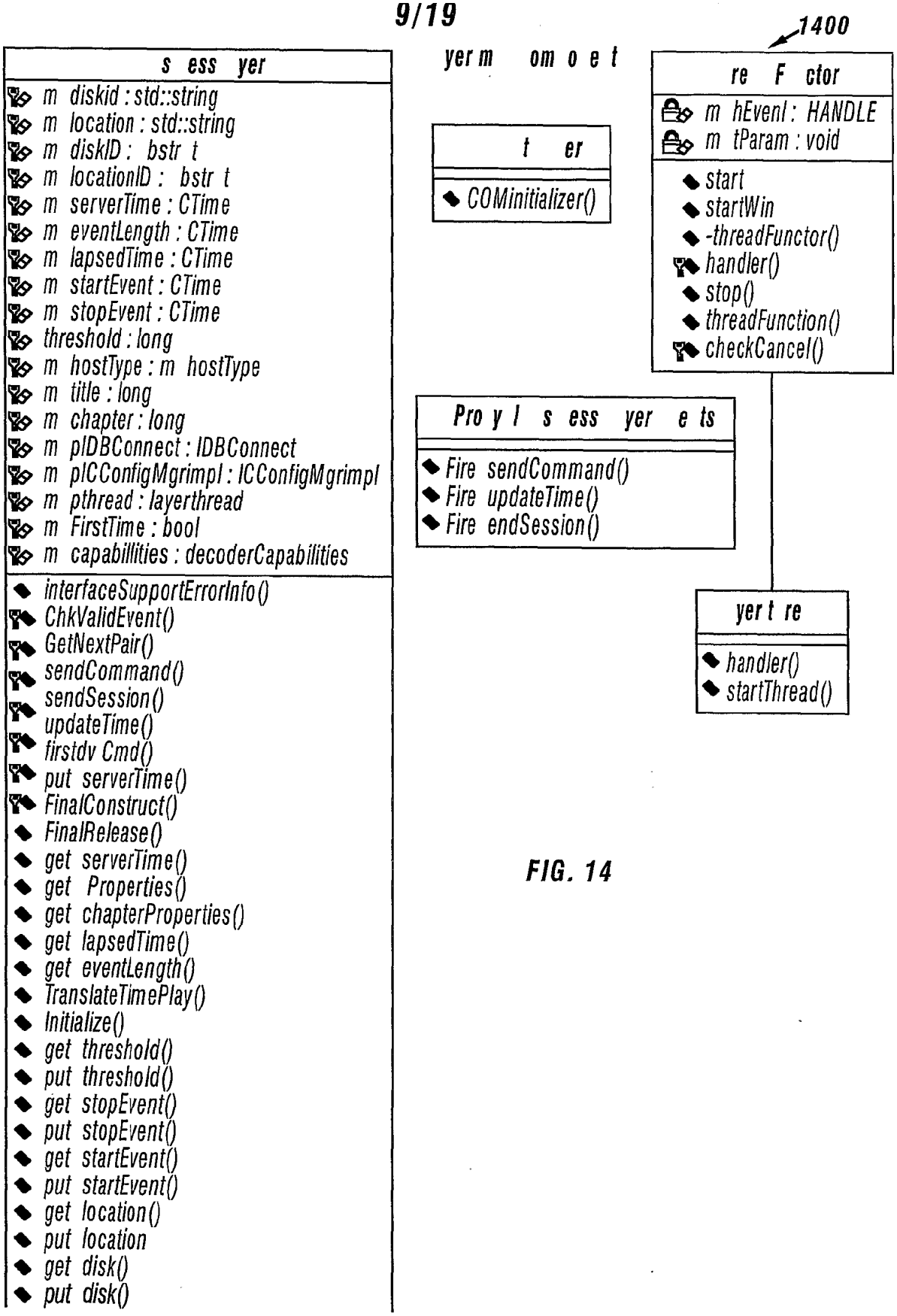

1400
yer m om o e t
s ess yer
m diskid : std::string
m location : std::string
m diskID : bstr t
m locationID : bstr t
m serverTime : CTime
m eventLength : CTime
m lapsedTime : CTime
m startEvent : CTime
m stopEvent : CTime
threshold : long
m hostType : m hostType
m title : long
m chapter : long
m pIDBConnect : IDBConnect
m pICConfigMgrimpl : ICConfigMgrimpl
m pthread : layerthread
m FirstTime : bool
m capabillities : decoderCapabilities
interfaceSupportErrorInfo()
ChkValidEvent()
GetNextPair()
sendCommand()
sendSession()
updateTime()
firstdv Cmd()
put serverTime()
FinalConstruct()
FinalRelease()
get serverTime()
get Properties()
get chapterProperties()
get lapsedTime()
get eventLength()
TranslateTimePlay()
Initialize()
get threshold()
put threshold()
get stopEvent()
put stopEvent()
get startEvent()
put startEvent()
get location()
put location
get disk()
put disk()
t er
COMinitializer()
re F ctor
m hEvenl : HANDLE
m tParam : void
start
startWin
-threadFunctor()
handler()
stop()
threadFunction()
checkCancel()
Pro y l s ess yer e ts
Fire sendCommand()
Fire updateTime()
Fire endSession()
yer t re
handler()
startThread()

FIG. 14

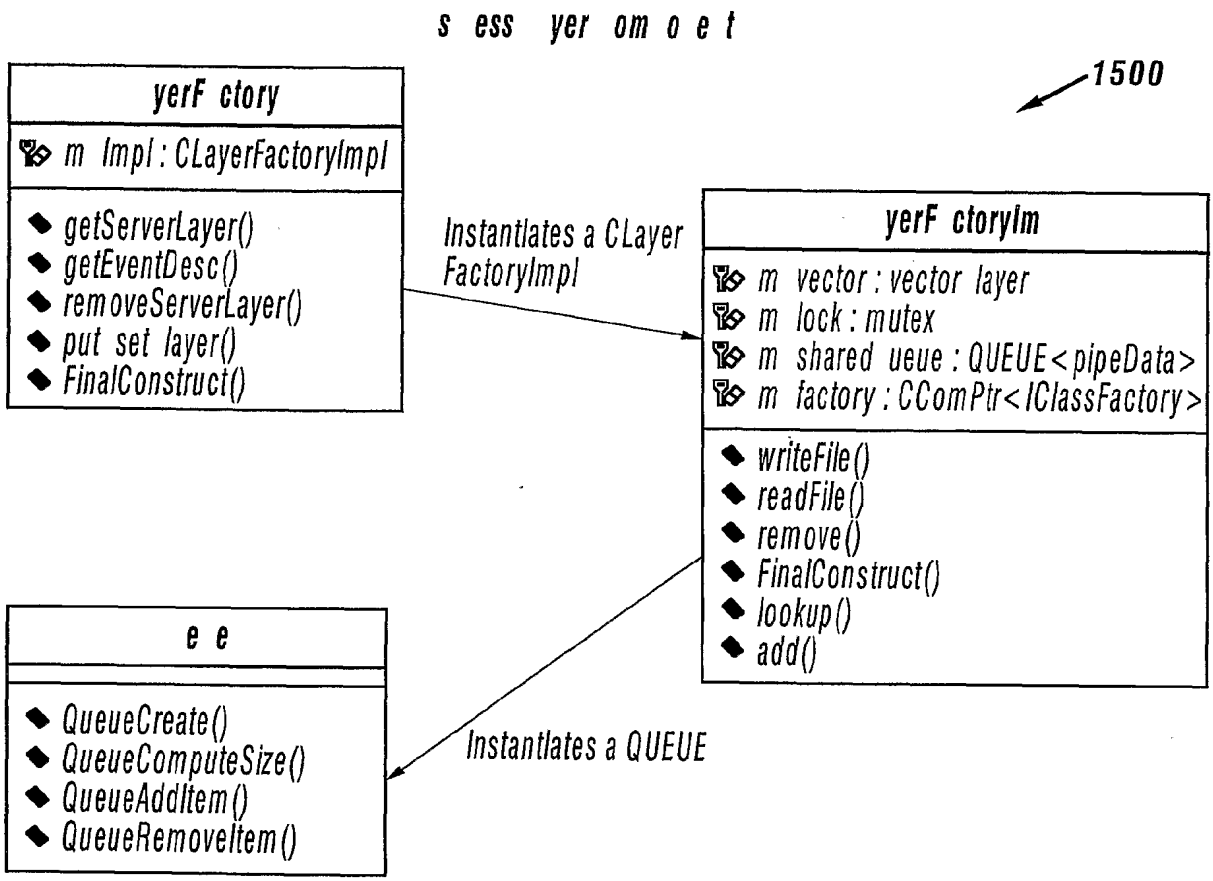

s ess yer om o e t
1500
yerF ctory
m Impl : CLayerFactoryImpl
getServerLayer()
getEventDesc()
removeServerLayer()
put set layer()
FinalConstruct()
Instantiates a CLayer FactoryImpl
yerF ctoryIm
m vector : vector layer
m lock : mutex
m shared ueue : QUEUE<pipeData>
m factory : CComPtr<IClassFactory>
writeFile()
readFile()
remove()
FinalConstruct()
lookup()
add()
e e
QueueCreate()
QueueComputeSize()
QueueAddItem()
QueueRemoveItem()
Instantiates a QUEUE

FIG. 15

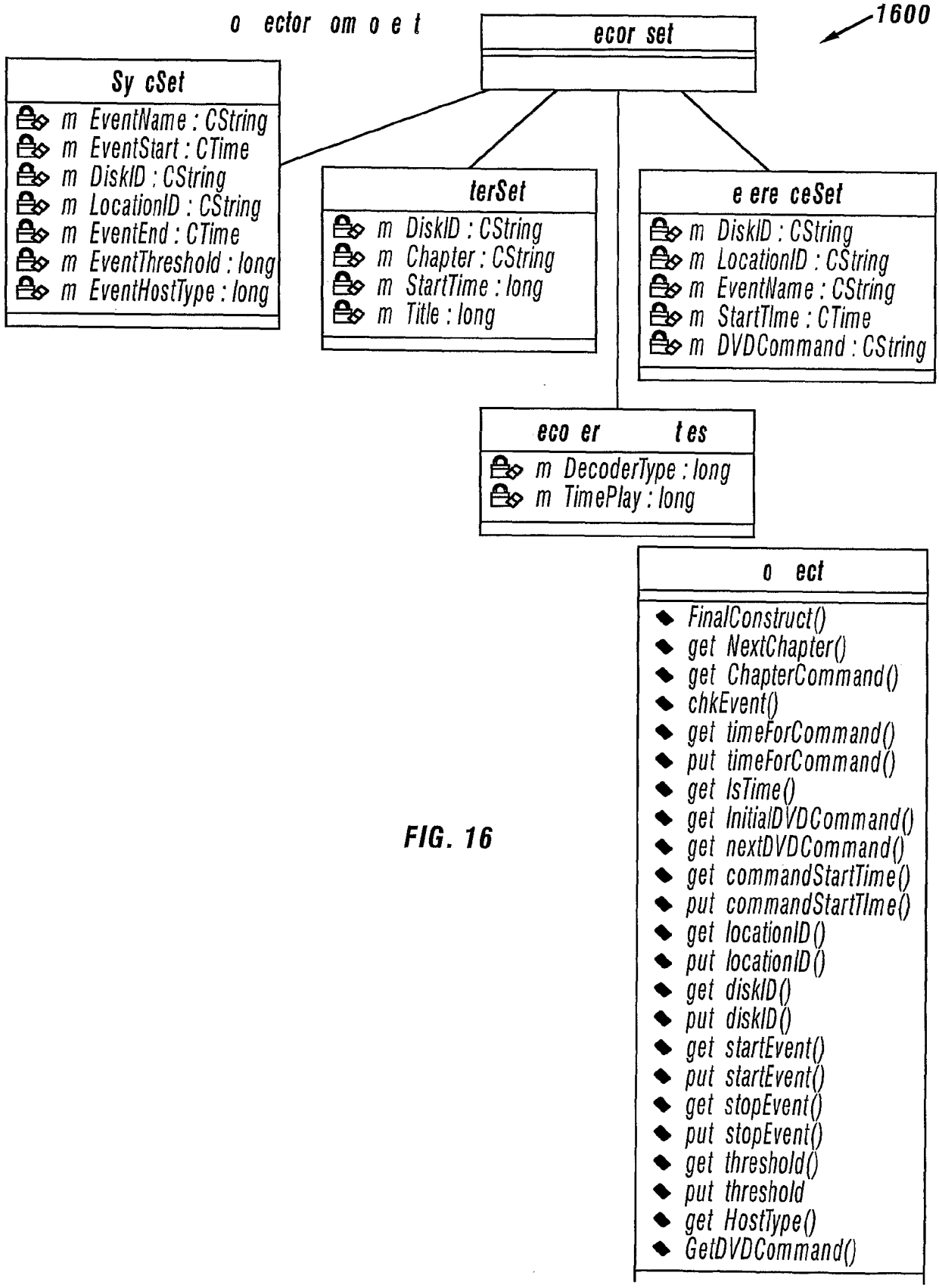

o ector om o e t
ecor set
1600
Sy cSet
m EventName : CString
m EventStart : CTime
m DiskID : CString
m LocationID : CString
m EventEnd : CTime
m EventThreshold : long
m EventHostType : long
terSet
m DiskID : CString
m Chapter : CString
m StartTime : long
m Title : long
e ere ceSet
m DiskID : CString
m LocationID : CString
m EventName : CString
m StartTIme : CTime
m DVDCommand : CString
eco er t es
m DecoderType : long
m TimePlay : long
o ect
FinalConstruct()
get NextChapter()
get ChapterCommand()
chkEvent()
get timeForCommand()
put timeForCommand()
get IsTime()
get InitialDVDCommand()
get nextDVDCommand()
get commandStartTime()
put commandStartTIme()
get locationID()
put locationID()
get diskID()
put diskID()
get startEvent()
put startEvent()
get stopEvent()
put stopEvent()
get threshold()
put threshold
get HostType()
GetDVDCommand()

FIG. 16

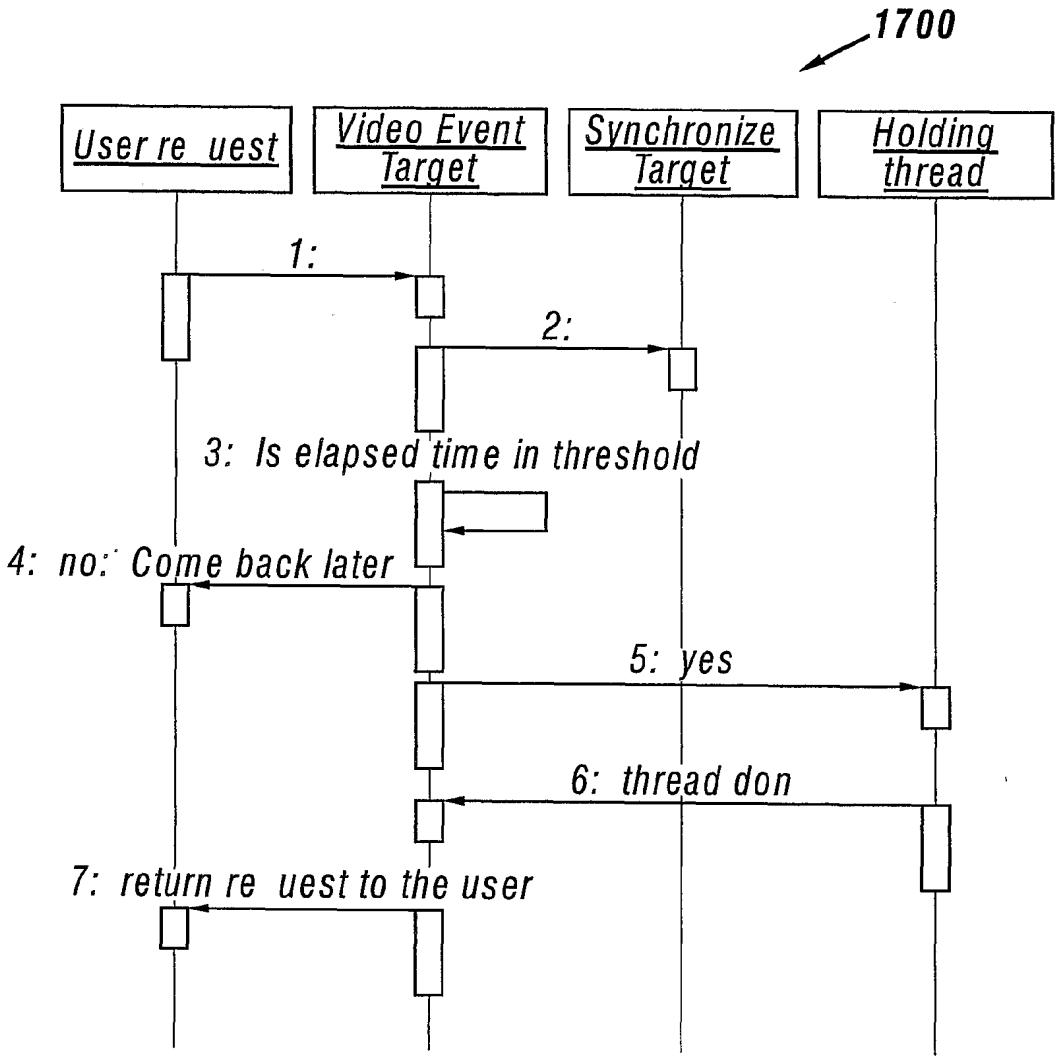

1700
User re uest
Video Event Target
Synchronize Target
Holding thread
1:
2:
3: Is elapsed time in threshold
4: no: Come back later
5: yes
6: thread don
7: return re uest to the user

FIG. 17

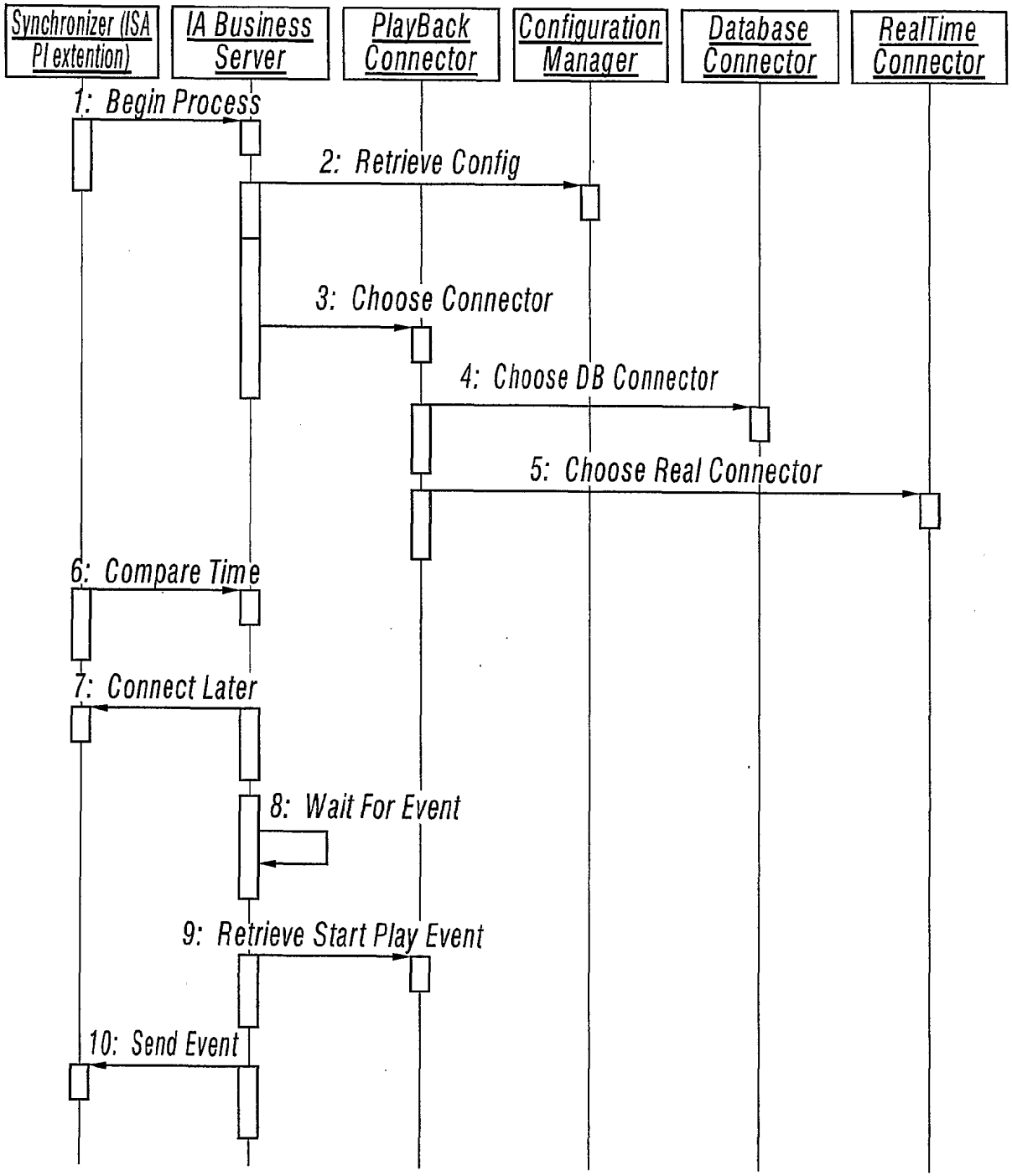

Synchronizer (ISA PI extention)
IA Business Server
PlayBack Connector
Configuration Manager
Database Connector
RealTime Connector
1: Begin Process
2: Retrieve Config
3: Choose Connector
4: Choose DB Connector
5: Choose Real Connector
6: Compare Time
7: Connect Later
8: Wait For Event
9: Retrieve Start Play Event
10: Send Event

FIG. 18

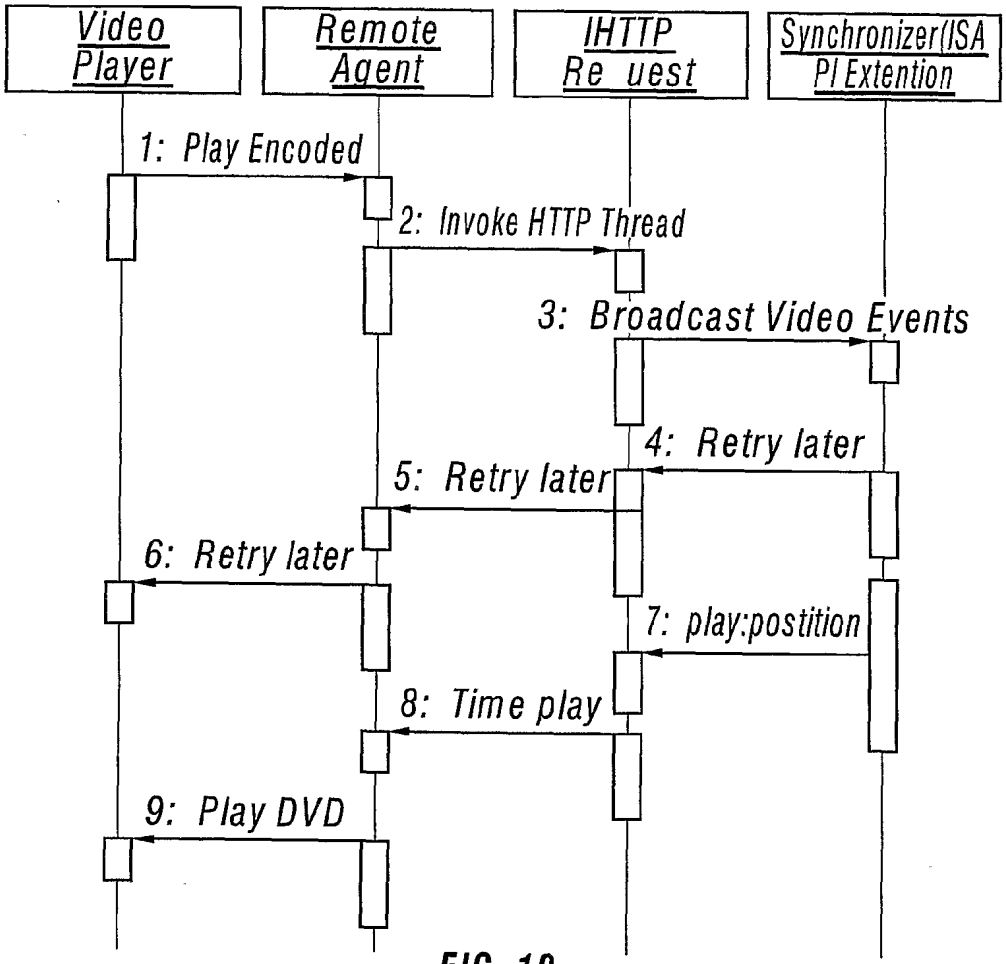
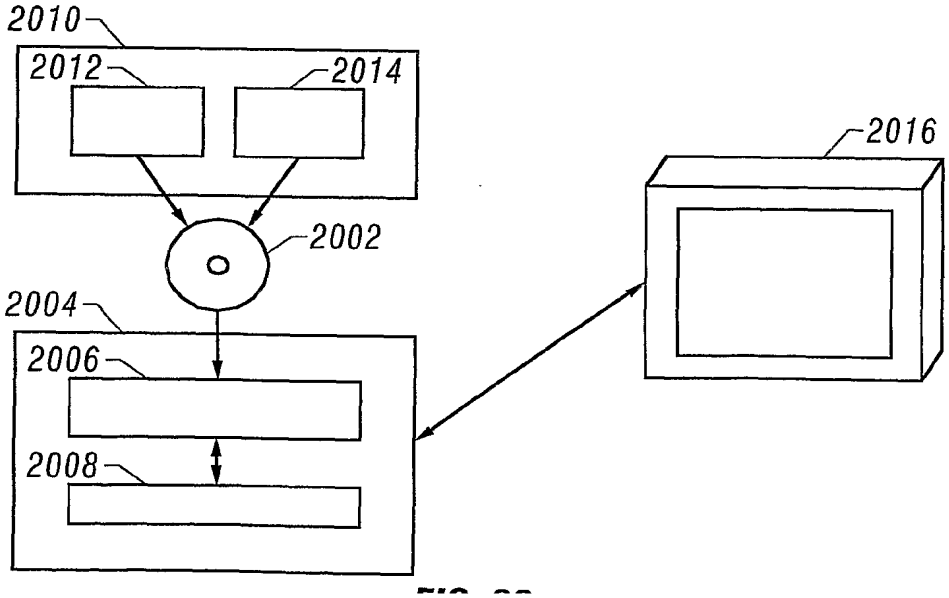

Video Player
Remote Agent
IHTTP Re uest
Synchronizer(ISA PI Extention
1: Play Encoded
2: Invoke HTTP Thread
3: Broadcast Video Events
4: Retry later
5: Retry later
6: Retry later
7: play:postition
8: Time play
9: Play DVD

FIG. 19


2010
2012
2014
2016
2002
2004
2006
2008

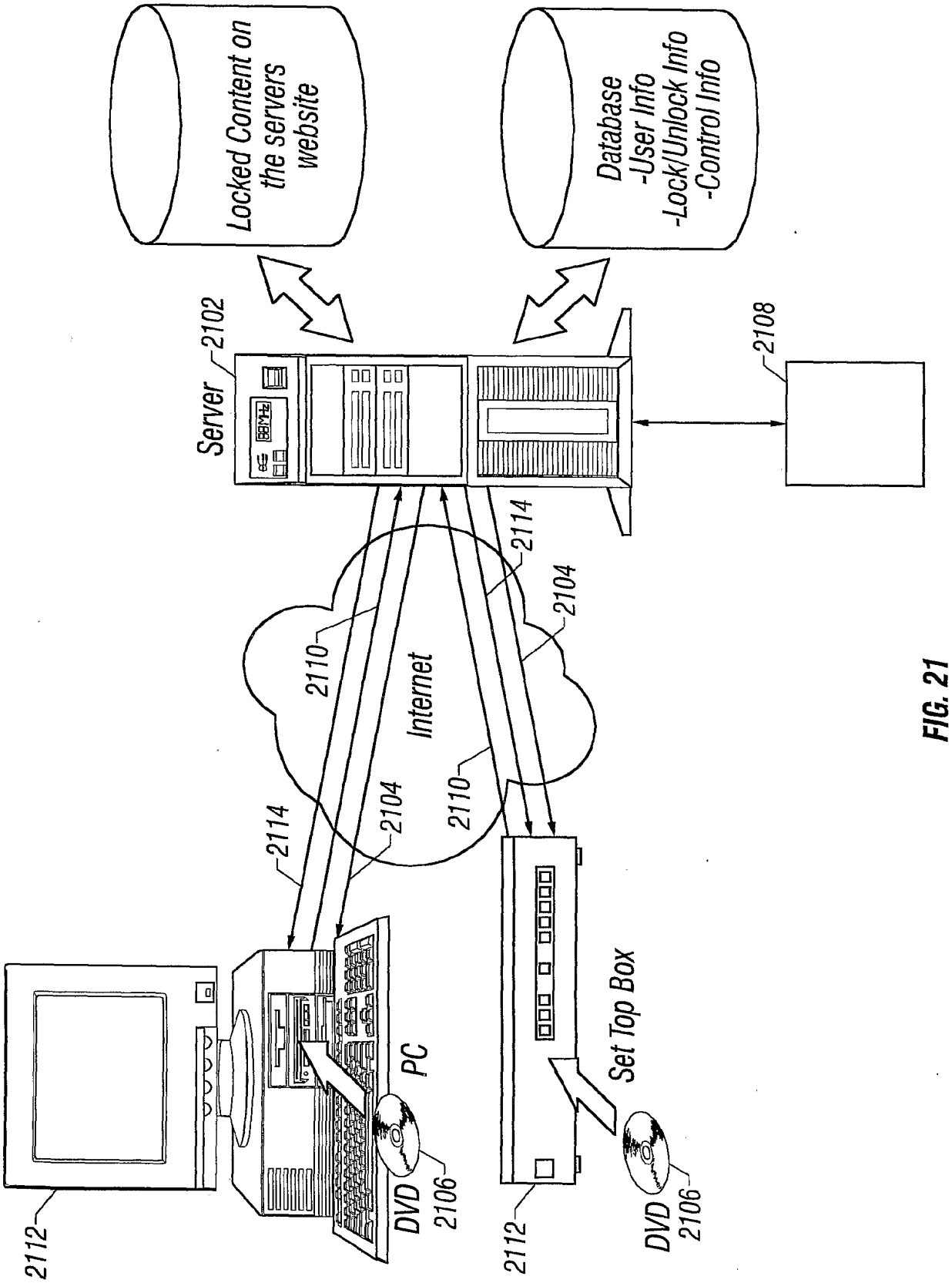

Locked Content on the servers website
Database
-User Info
-Lock/Unlock Info
-Control Info
Server
2102
2108
2114
2104
2110
Internet
2110
2104
2114
PC
Set Top Box
DVD
2106
DVD
2106
2112
2112

FIG. 21

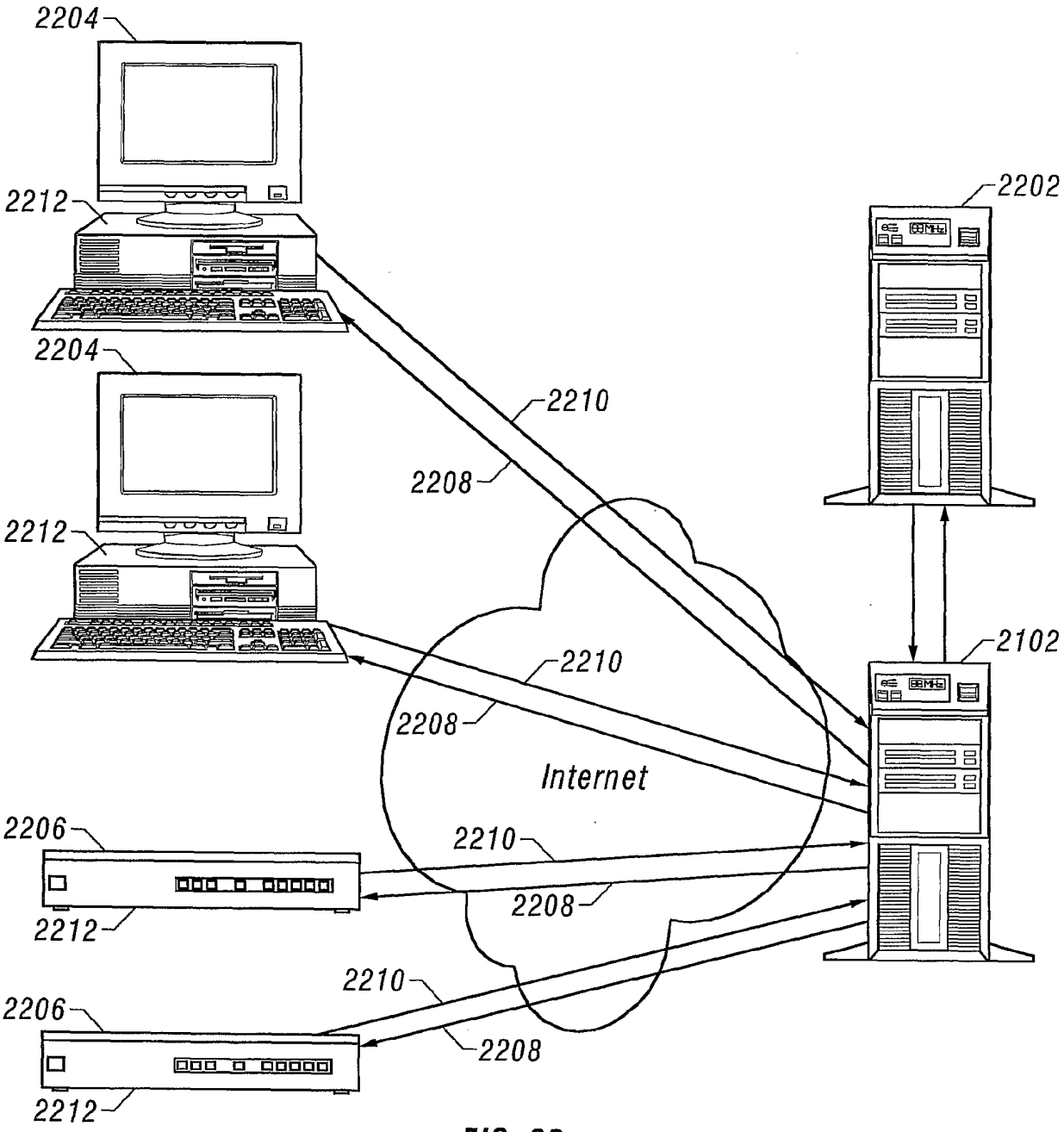

2204
2212
2204
2212
2210
2208
2202
2102
2210
2208
Internet
2206
2210
2212
2208
2210
2206
2208
2212

FIG. 22

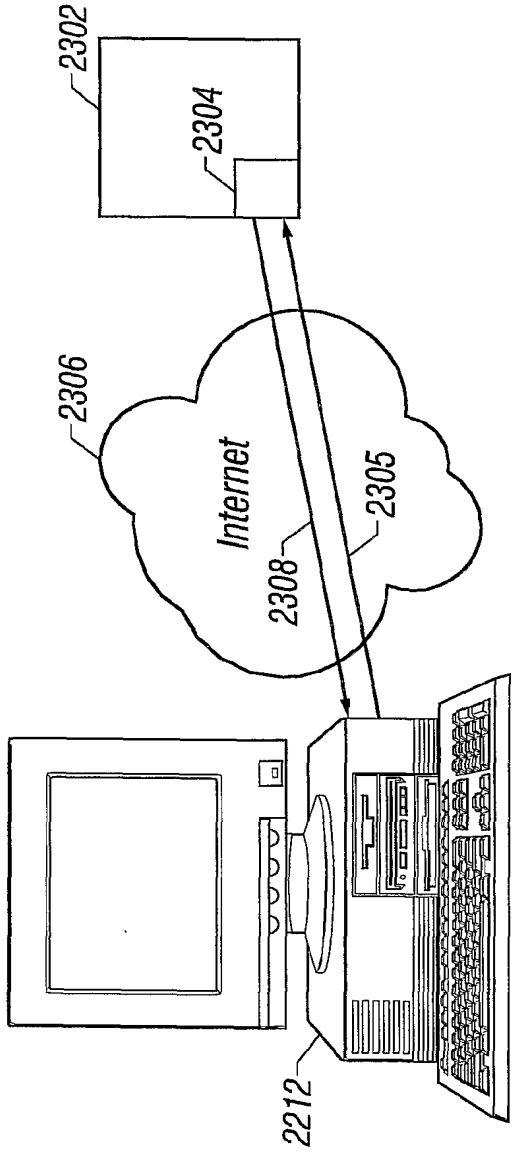


FIG. 23

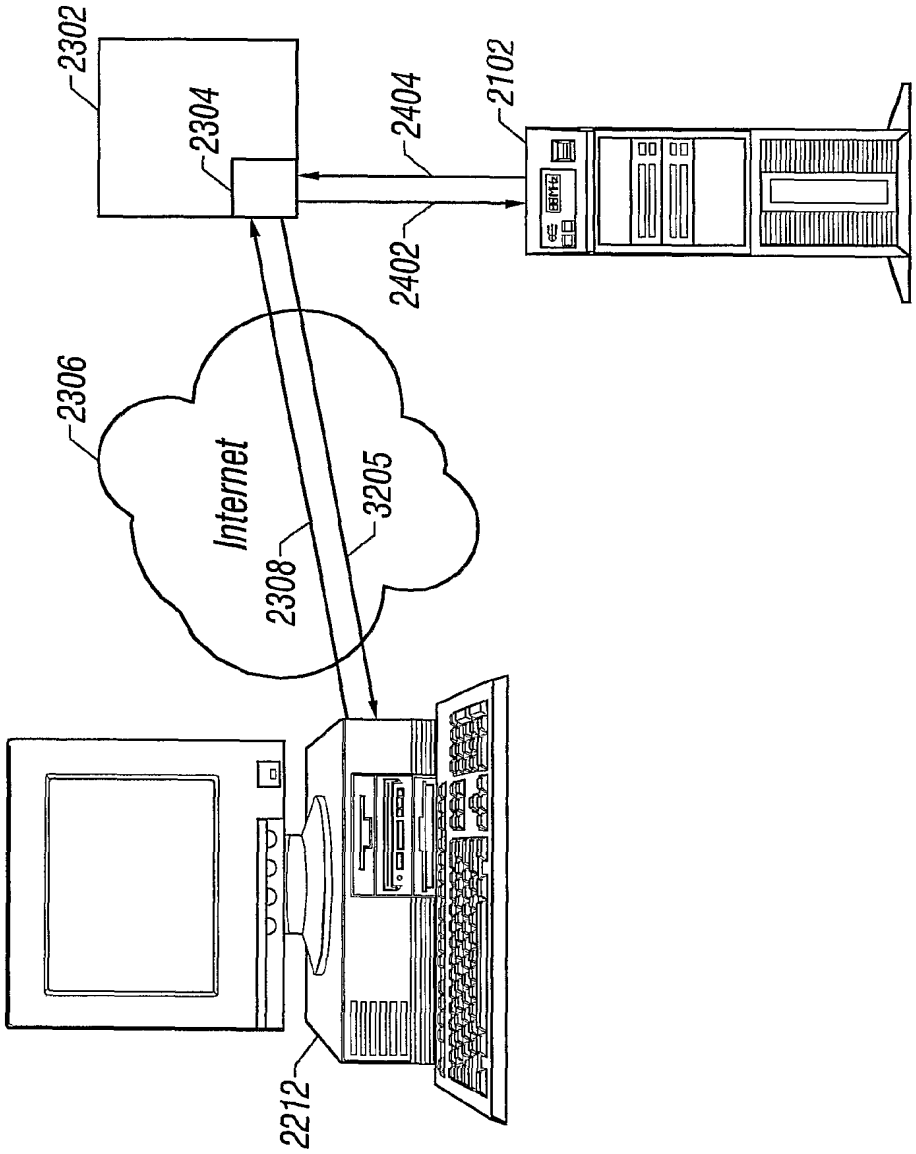


FIG. 24

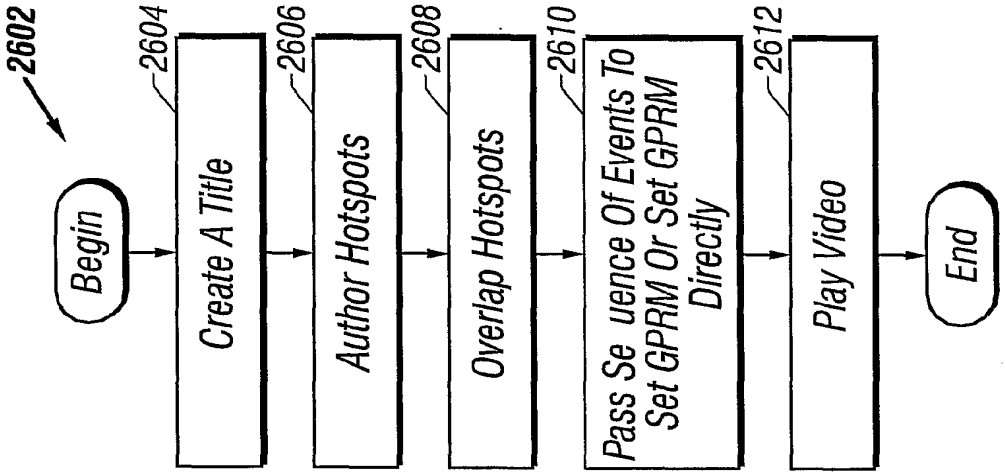

2602
Begin
2604
Create A Title
2606
Author Hotspots
2608
Overlap Hotspots
2610
Pass Se uence Of Events To Set GPRM Or Set GPRM Directly
2612
Play Video
End

FIG. 26

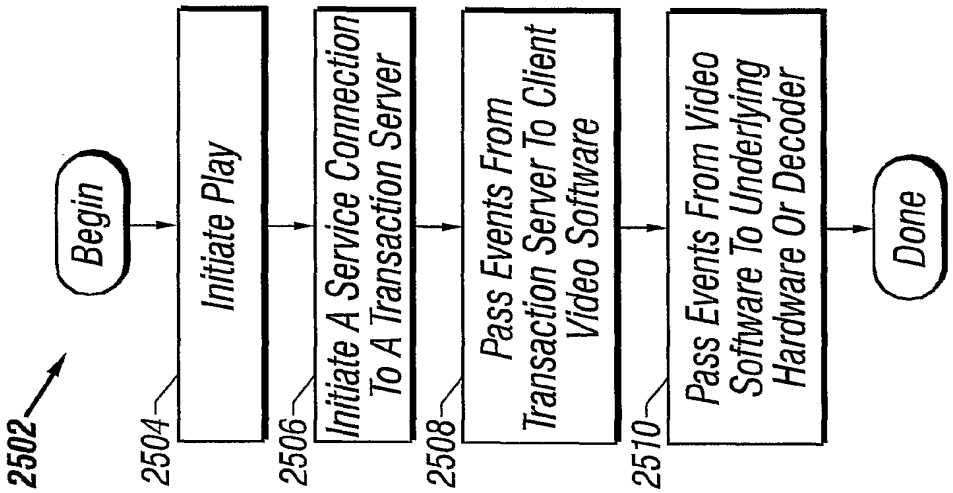

2502
Begin
2504
Initiate Play
2506
Initiate A Service Connection To A Transaction Server
2508
Pass Events From Transaction Server To Client Video Software
2510
Pass Events From Video Software To Underlying Hardware Or Decoder
Done

FIG. 25

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/02143

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :H04L 9/00; G06F 13/00; H04L 9/00, 9/08, 9/32
US CL :380/203, 277; 709/217, 218

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 380/203, 277; 709/217, 218

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 6,035,329 A (MAGES ET AL.) 07 MARCH 2000, COL.2-5.	1-185
Y, P	US 6,185,306 B1 (MAGES ET AL.) 06 FEBRUARY 2001, COL.3-10.	1-185
Y	US 5,937,164 A (MAGES ET AL.) 10 AUGUST 1999, COL.3-8.	1-185
Y	US 5,892,825 A (MAGES ET AL.) 06 APRIL 1999, COL.3-8.	1-185

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
02 APRIL 2001	26 APR 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer GLEN BURGESS James R. Matthews Telephone No. (703) 305-4792

Form PCT/ISA/210 (second sheet) (July 1998)★